1985

Integration of a company's engineering and manufacturing departments utilizing CAD/CAM technologies /

Lev Nelik
Lehigh University

"INTEGRATION OF A COMPANY'S ENGINEERING AND MANUFACTURING DEPARTMENTS UTILIZING CAD/CAM TECHNOLOGIES"

by

Lev Nelik

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Manufacturing Systems Engineering

Lehigh University,

Bethlehem, Pa.

1985

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

11/11/85

(Date)

Tulga Ozsoy

Professor in Charge

Roger N Nagel

Director of MSE Program

F. Erdogan

Chairman of M.E. & Mech. Dept.

TABLE OF CONTENTS

TABLE OF CONTENTS

1. ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Tulga Ozsoy, at Lehigh University, for his guidance throughout my work on this thesis, and for numerous informative lectures and discussions on relevant topics.

Also, I wish to thank Peter D'Souza, Manager of Manufacturing for Ingersoll-Rand Company of Phillipsburg, New Jersey, who has been most instrumental in setting the direction and goals for this work. I would also like to thank Barry Cimino, Chuck Morgan, John O'Connell, and Gerry Gormley, all from Ingersoll-Rand, who have been very instrumental in helping to accompish this task.

Finally, I would like to thank my wife, Wendy, for her patience and assistance during my year as a graduate student.

2. ABSTRACT

The objective of this thesis was to establish a computerized information transfer procedure between the Engineering and Industrial Engineering departments at the Ingersoll-Rand Company plant at Phillipsburg, New Jersey.

As a result of the analysis of the company's interdepartmental comminications, the following was accomplished:

A. A Conceptual Diagram of the Order Processing in the Engineered Pump Division was generated.

B. Parametric computer programs to design pump parts and display geometrical representations on the computer graphical terminal and subsequent database processing for the purpose of generating computer files used by the NC machines were written.

C. A computer network link between the Engineering and Industrial Engineering computer mainframes for the purpose of the NC files transformations was established.

3. INTRODUCTION

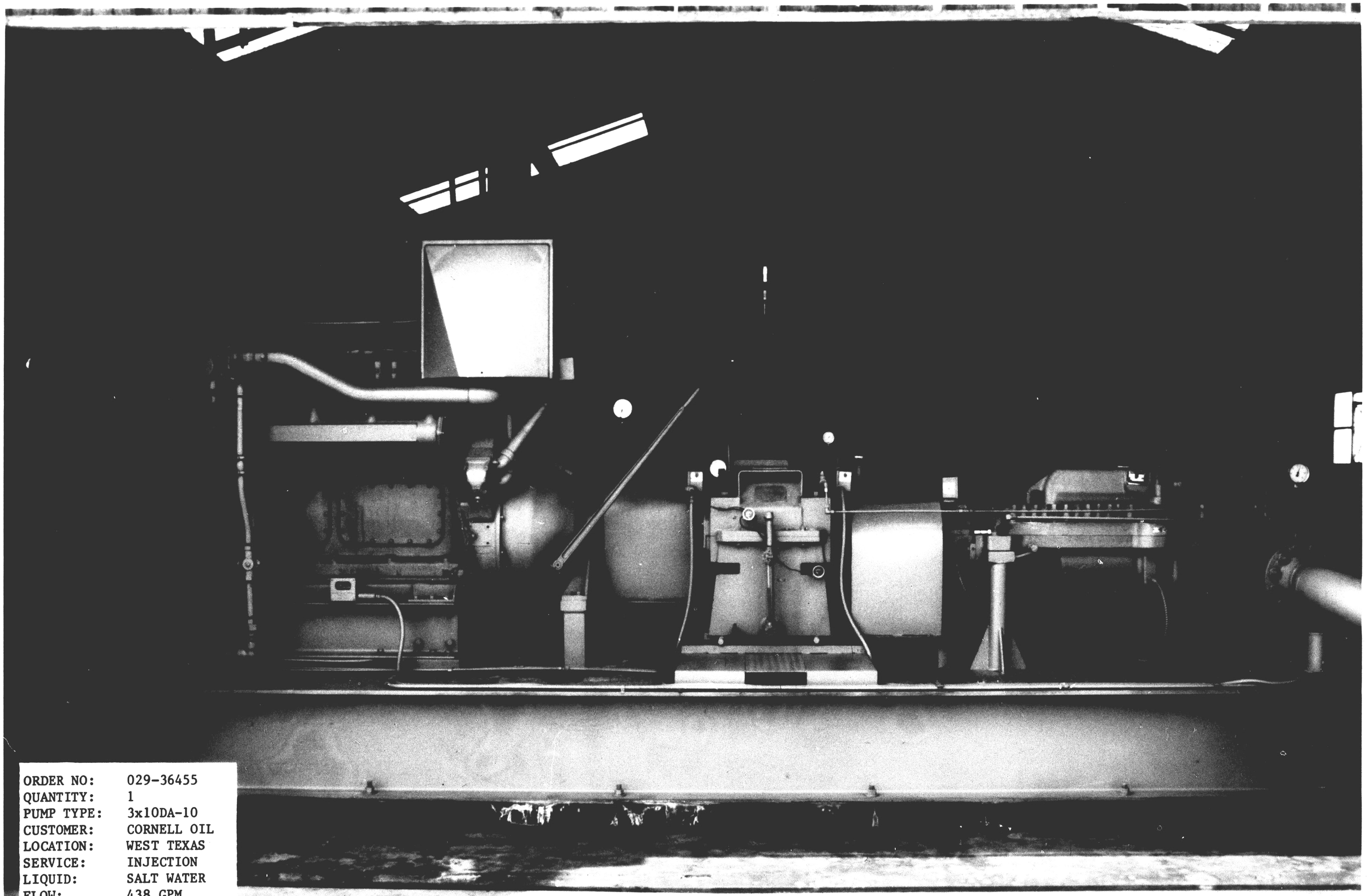
The work was performed at the Ingersoll-Rand Company, Engineered Pump Division located in Phillipsburg, New Jersey.

Centrifugal pumps are the main product that is designed and manufactured by this Division. An example of a pump used for the pipeline service is shown on Fig. 1. Fig. 2 shows a complete pump/driver assembly used for salt water injection. There are several lines of pumps manufactured by Ingersoll-Rand. The current work addressed a specific pump type (DA-line). During the last several years there has been a dramatic change in the Division operations. The introduction of the Computer-Aided-Design-and-Manufacturing technologies has opened a new horizon to effectively use these technologies to decrease cost, lead time and improve product.

To utilize new technologies most effectively, the overall Division's operations needed to be reevaluated using the Manufacturing System's approach. The work of each department and communications between the departments were examined. A Conceptual Diagram of Engineered Pump Division Order Processing was developed.

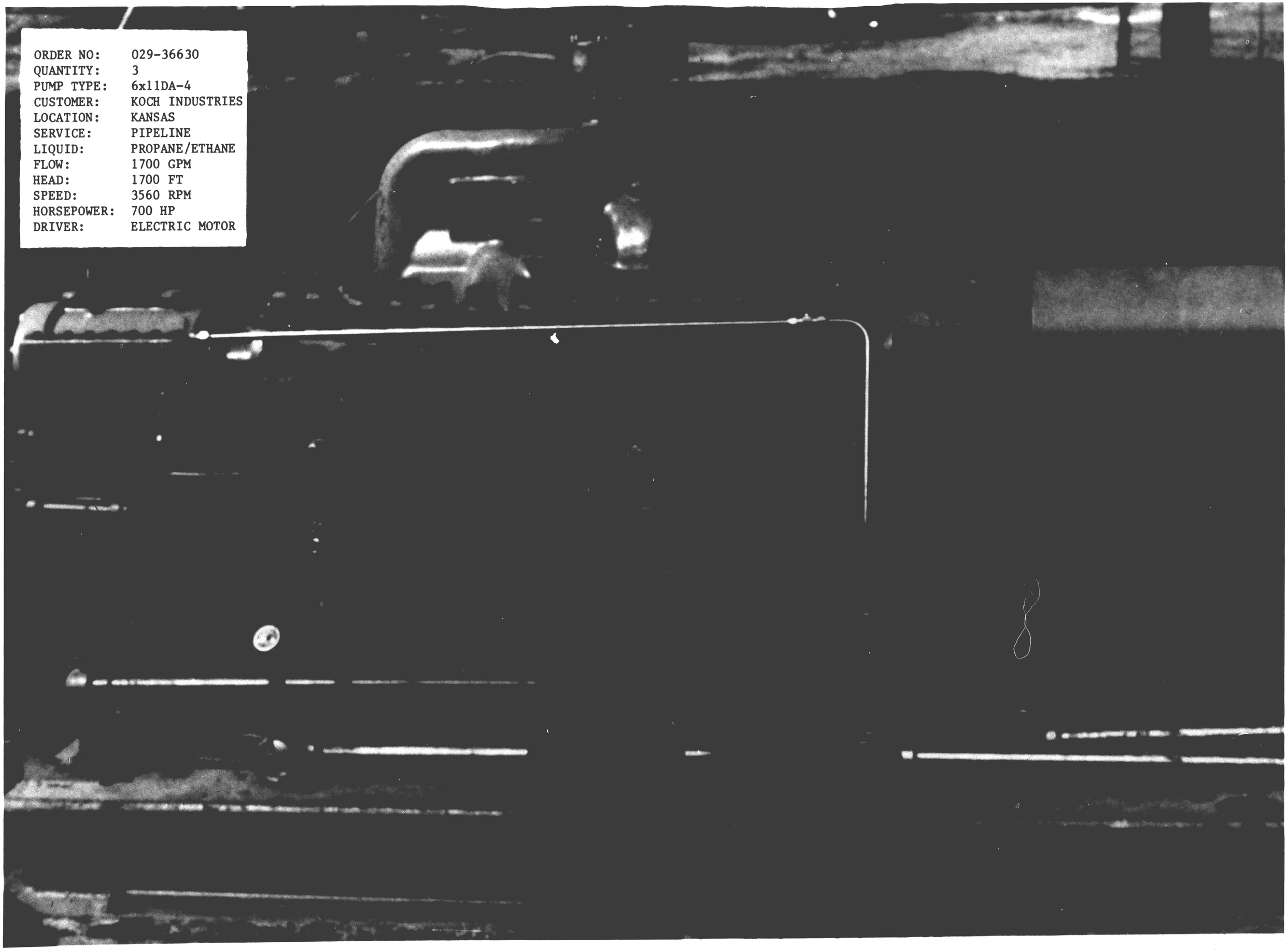
Then the potential areas for improvement through the introduction of CAD/CAM and other technologies became obvious. The diagram became a dynamic tool, changing with the requirements. It is currently used by the management to guide the overall Division's plan of Automation and Computer Assisted Operations.

The following sections describe the conventional and the newly introduced flows of Order Processing and elaborate specifically on what was done in the areas of CAD/CAM.



ORDER NO:	029-36455
QUANTITY:	1
PUMP TYPE:	3x10DA-10
CUSTOMER:	CORNELL OIL
LOCATION:	WEST TEXAS
SERVICE:	INJECTION
LIQUID:	SALT WATER
FLOW:	438 GPM
HEAD:	3555 FT
SPEED:	3950 RPM
HORSEPOWER:	1000 HP
DRIVER:	DIESEL ENGINE

ORDER NO:	029-36630
QUANTITY:	3
PUMP TYPE:	6x11DA-4
CUSTOMER:	KOCH INDUSTRIES
LOCATION:	KANSAS
SERVICE:	PIPELINE
LIQUID:	PROPANE/ETHANE
FLOW:	1700 GPM
HEAD:	1700 FT
SPEED:	3560 RPM
HORSEPOWER:	700 HP
DRIVER:	ELECTRIC MOTOR



4. CONVENTIONAL ORDER PROCESSING

Figure 3 shows the overall diagram of how the order is processed through the system from the moment it enters the system by the customer to the point when the complete unit is shipped to the customer.

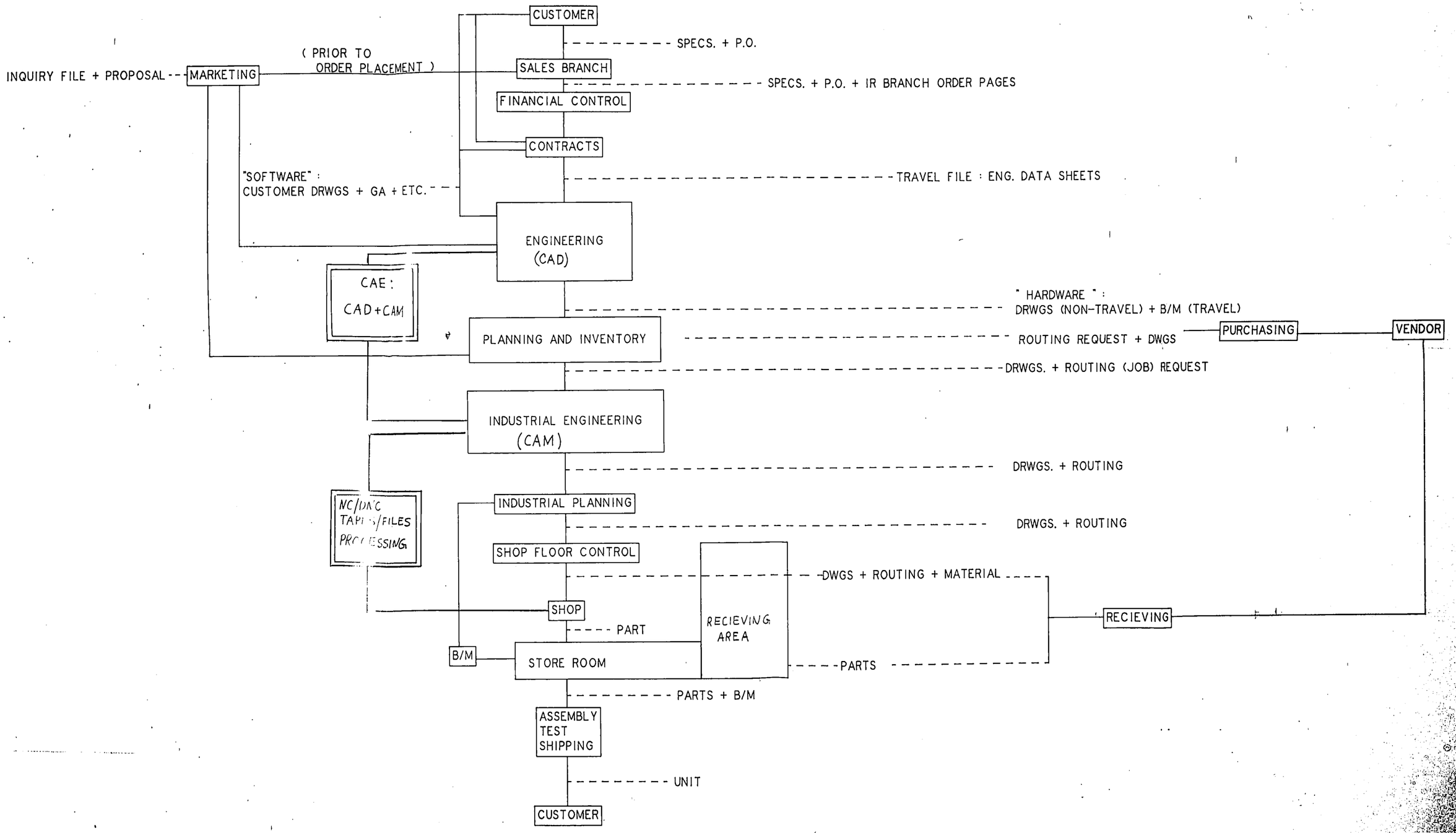
4.1 Sales Branch

The Sales Branch is a company's front end. In essentially all cases, the customer's inquiry on a particular pump design enters the System through one of the many company's Sales Branches located throughout the world.

Sales Branches receive customer's specifications and an inquiry letter or a telephone inquiry and submit this information to Marketing. After the inquiry becomes an order, Sales generate Branch Order Pages which, together with the Specifications and Purchase Order, is submitted to Financial Control.

FIG. 3

CONCEPTUAL DIAGRAM
OF ENGINEERED PUMP DIVISION
ORDER PROCESSING
(CONVENTIONAL)



4.2 Marketing

Marketing performs most of the preliminary work with the inquiry until an inquiry becomes an order. Marketing is responsible for obtaining information from Engineering about specific technical details of the job and from Planning and Inventory about the materials availability to meet the required delivery dates.

After obtaining all information that is required, Marketing submits a proposal to the customer. If the customer finds the proposal acceptable and compares with similar proposals from other pump manufacturers, he places an order.

4.3 Financial Control

Financial Control receives the information from Sales and performs all of the required financial operations with it. Financial Control also works closely with Marketing during the inquiry stage.

4.4 Contracts

Upon approval of the order by Financial Control, the Contracts Department initiates a Travel file which contains Engineering Data Sheets which need to be filled out by Engineering and returned to Contracts. Contracts will send some of the information from the completed Engineering Data Sheets to the customer either directly (if specified in the Order Pages) or through the Sales Branch.

4.5 Engineering

Engineering has all responsibilities of verifying a selection of the existing design or makes a new design. Engineering performs all required analysis, such as strength, performance, etc. and generates a complete set of drawings. Some of these drawings, known as a "software", is sent to the customer; these include General Arrangement layouts, piping layouts, etc., and the others, known as "hardware", goes to Planning and Inventory. These include the drawings of all parts that are needed to be manufactured or purchased. Engineering also makes a Bill of Materials, which include complete listing of parts and assemblies that make up a pump unit. Bill of Materials is also called a "Travel file" since it is used by all subsequent departments.

Engineering represents a prime potential area of a CAD/CAM implementations. Engineering function will be analysed in greater detail in Section IV.

4.6 Planning and Inventory

Planning and Inventory has a responsibility of Scheduling materials deliveries and smooth order processing through the operations. This department generates a Routing Request which is submitted to Industrial Engineering to generate a routing for the shop.

Planning and Inventory is in direct contact with vendors.

4.7 Industrial Engineering

Industrial Engineering is responsible for all of the operations that follow it. Industrial Engineering generates a routing based on the Routing Request that is received from Planning and Inventory. All NC programming, machining time study, tooling and so on, is done by the Industrial Engineering Department.

As the Engineering Department, Industrial Engineering has a big potential for the CAD/CAM implementations. Industrial Engineering function will be analysed in greater detail in Section V.

4.8 Industrial Planning

Industrial Planning receives information from Industrial Engineering, verifies it and submits to the Shop Floor Control. Industrial Planning also verifies the Bill of Materials and sends it to the Storeroom.

4.9 Shop Floor Control

Shop Floor Control obtains the material from the Storeroom and, together with Industrial Planning, allocates Shop machinery for the required operations.

4.10 . Shop

Shop is a place where the parts are made. The overall Systems efficiency is truely tested here. The successful shop operations depend on efficient work of all of the preceding departments and all mistakes also show up here.

4.11 Storeroom

Storeroom maintains inventories. It receives materials and parts from the vendors and from the Shop. Storeroom verifies the availability of parts and subassemblies in accordance with the Bill of Materials which it receives from the Industrial Planning. When all parts that make up a complete unit are in the Storeroom, it submits them to the Assembly Floor.

4.12 Assembly, Test, Shipping

These stages conclude the operations of the Order Processing. The complete unit is then shipped to the customer. After that the unit becomes the responsibility of the Contract Department, which will provide maintenance work, parts scheduled replacement and will maintain overall history of operation of the installed product.

5. ENGINEERING FUNCTION

5.1 System Description

In the previous section, the function of Engineering was described in relation with other functions of the System. In this section more detailed description of the Engineering function will be given taking a DA-line pump impeller design and manufacturing process as an example..

As was mentioned earlier, Engineering receives a Travel File from the Contracts. A Travel File also includes a Master Schedule. A Master Schedule contains a listing of standard pump parts and the new parts that have to be designed. Engineering will design the necessary parts according to the Master Schedule. The need for either casting or machining drawing (or both) is specified in the master schedule. Figure 4 shows a part (DA-Line pump impeller) design process. A computer-aided-design (CAD) has been implemented for the design of the pump parts and to generate required drawings. The mainframe computer used by Engineering is a VAX 751 with 7 graphical workstations ("Applicon") and 3 VT-220 terminals. "Applicon" software is used for the graphical work.

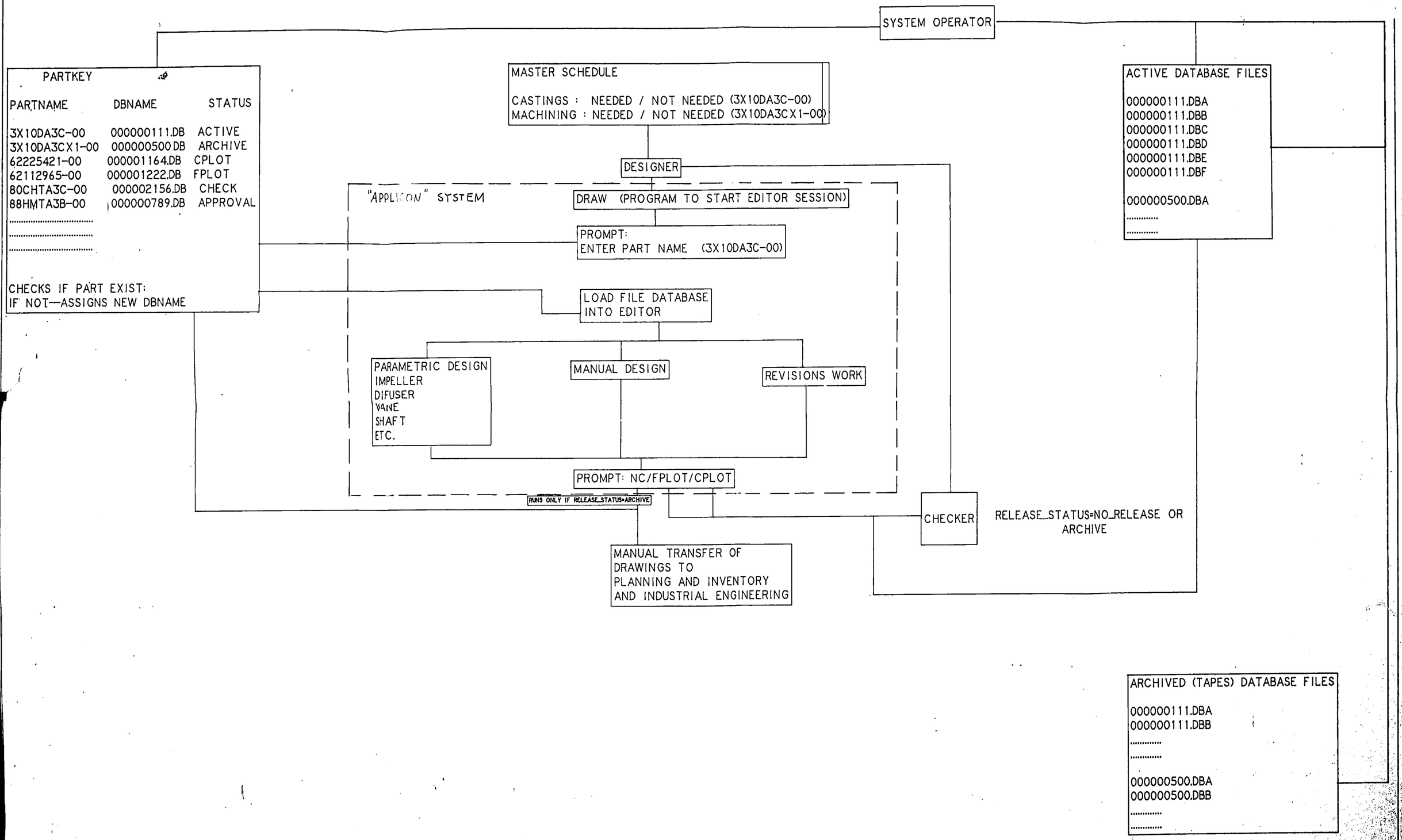
The computer database file management is performed using DEC file management system:

1. Active database files (controlled by "Applicon" database manager)
2. Archived database files (on tapes)
3. "PARTKEY" Manager (written in-house)

This is shown on Figure 4.

FIG. 4

ENGINEERING DESIGN.
(CONVENTIONAL)



A design can be done either manually or using parametric programs which are developed in-house for the design of certain parts. These programs are written in IAGL (Interactive Applicon Graphics language). Also, revision work can be perfomed to change or modify existing designs. Some of the parametric programs that are currently available are:

"VANE" - used for a hydraulic design of a pump or turbine impeller vanes

"DIFFUSOR" - for hydraulic design of pump diffusor vanes

"SHAFT" - to design pump shafts and others.

5.2 The Management of the Engineering Database Files

In order to efficiently usecomputer resourses, only a limited information is stored on the computer. This information is immediately accessible. The work that requires only occasional access is stored on tapes and the files are taken off the computer. The File Manager is a special program written in DCL (DEC Control language), which keeps track of the files status. The files status can be "NO_RELEASE" or "ARCHIVE". The following is a description of the System file status activities. When the designer complete the initial hydraulic design, he stores the drawing (i.e. database file) on the system as an active database file and generates a paper plot ("CPLOT = "CHECK_PLOT" or "FPLOT" = "FINAL PLOT"). The plot is then received by the checker who approves it and sends the "RELEASE_STATUS" = "ARCHIVE". If changes are required at that step or there are any errors, the "RELEASE_STATUS" remains set to "NO_RELEASE". The designer then receives the marked-up drawing and makes the necessary changes. The process is repeated until the work is approved and the "RELEASE_STATUS" is set to "ARCHIVE".

The File Manager also sends requests to the System operator to move the files fromthe computer memory or store/retrieve files from tapes.

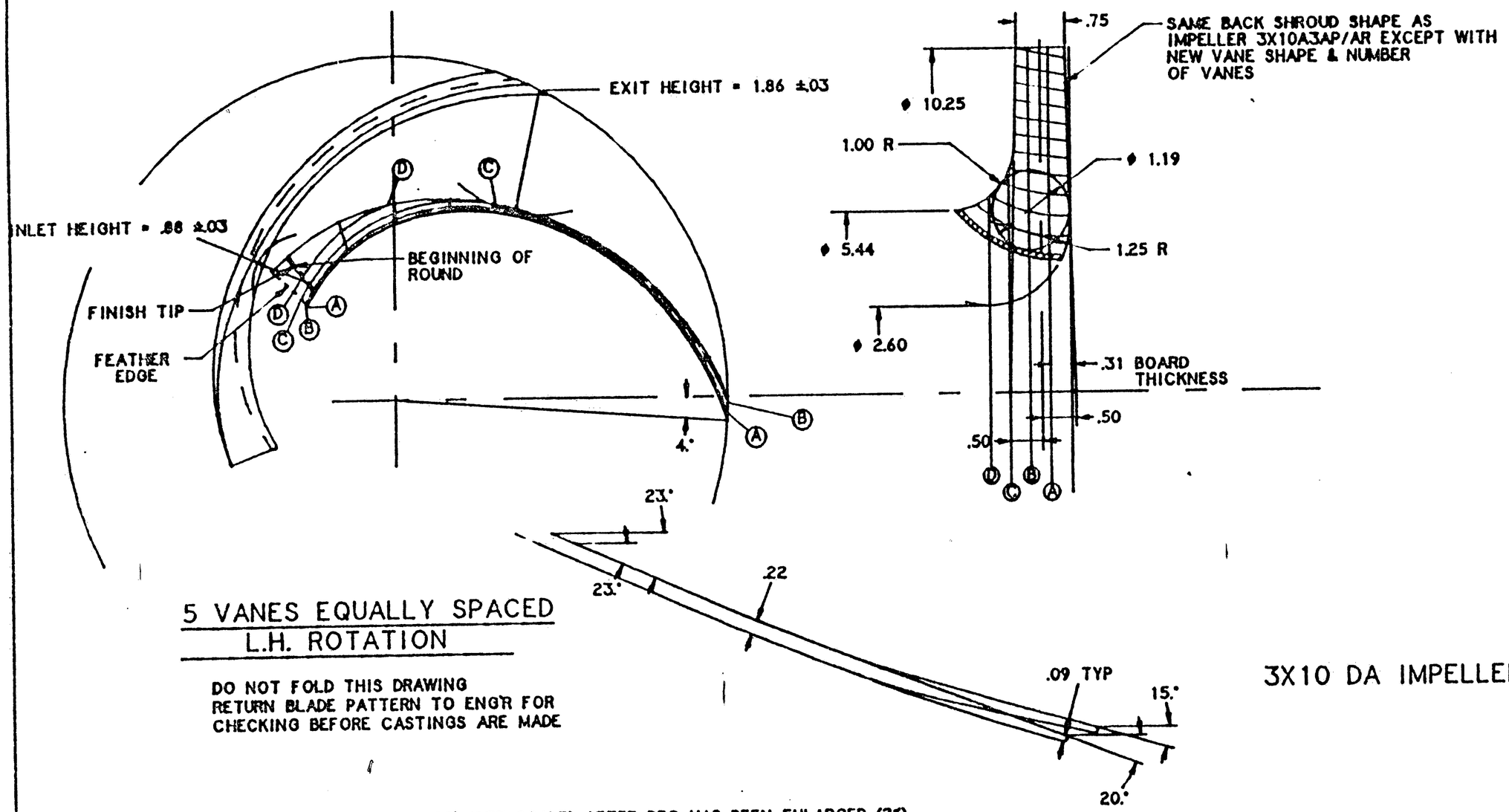
The name of the file ("PARTNAME") can be of any length, but the operating system allows only 9-digit identification number. PARTKEY" gets around this problem by assigning a special 9-digit "DBNAME" to the files and keeps track of them.

5.3 Operation Under "Applicon" Editor

Editor is a graphical computer software written by "Applicon". Its purpose is to accept graphical information, store it in a database format, and maintain an interactive dialog with the user.

A designer enters the Editor by typing "DRAW" followed by the part name (e.g. 3X10DA3C-00). The "DRAW" program uses a "PARTKEY" that has the database (i.e. drawing) with the "DBNAME" that has the required "PARTNAME" identification "key". In the example shown on Figure 4, the program will find and load to the "Editor" the file with the "DBNAME"= 000000111.DB; since it contains a required key ("PARTNAME" = 3X10DA3C-00).

The file will be loaded to the scratch area allocated by the "Editor" immediately, since the "PARTKEY STATUS" = "ACTIVE", i.e. it is on the disk. If the "RELEASE_STATUS"= "ARCHIVE", the system operator would have to load the file to the disk from the tape. If the required file does not exist, a computer will assign a new incremental "DBNAME" for it and load an empty file to the scratch area. This file will have a "PARTNAME" = 3X10DA3C-00 as entered by the designer. The design work then begins.



5 VANES EQUALLY SPACED
L.H. ROTATION

DO NOT FOLD THIS DRAWING
RETURN BLADE PATTERN TO ENG'R FOR
CHECKING BEFORE CASTINGS ARE MADE

THIS LINE TO MEASURE 12.25" (STD SCALE) AFTER DRG HAS BEEN ENLARGED (2x)

THIS LINE TO MEASURE 12.19" (STD SCALE) AFTER DRG HAS BEEN ENLARGED (1-1/2x)

3X10 DA IMPELLER

3X10DA3ANX0-00

Fig. 5 - Impeller Hydraulic Layout

5.4 Drawing Data Management

Following the completion of hydraulic design, the impeller machining and casting drawings need to be generated. The hydraulic design database file is retrieved from the database and the designer makes a machining drawing. Initially, the outside contours of the impeller are established (Figure 6). At this stage other pump parts such as casing (Figure 7) are retrieved from the database to verify if there are any interferences between the casing and the impeller. When this is done, the dimensioning of the impeller is performed (Figure 8). The designer then stores the drawing as a database file and generates a paper plot. The paper plot is sent to the Checker for approval. The design steps at this point are similar to those taken for the hydraulic layout. When the final paper drawing is approved, it is then ready to be released to Planning and Inventory and Industrial Engineering.

Following the completion of the machining drawing, the impeller casting drawing must be made (for the cast impellers). Often, though, this work is performed in parallel with the generation of the machining drawing.

Essentially, a casting drawing represents a machining drawing with the added tolerances for the metal to be removed from the casting by machining (Figure 9).

This drawing will be required by:

A. Foundry

B. Industrial Engineering (to determine the tool offsets when machining the part)

The casting drawing design procedure is similar to the design of the machining drawing.

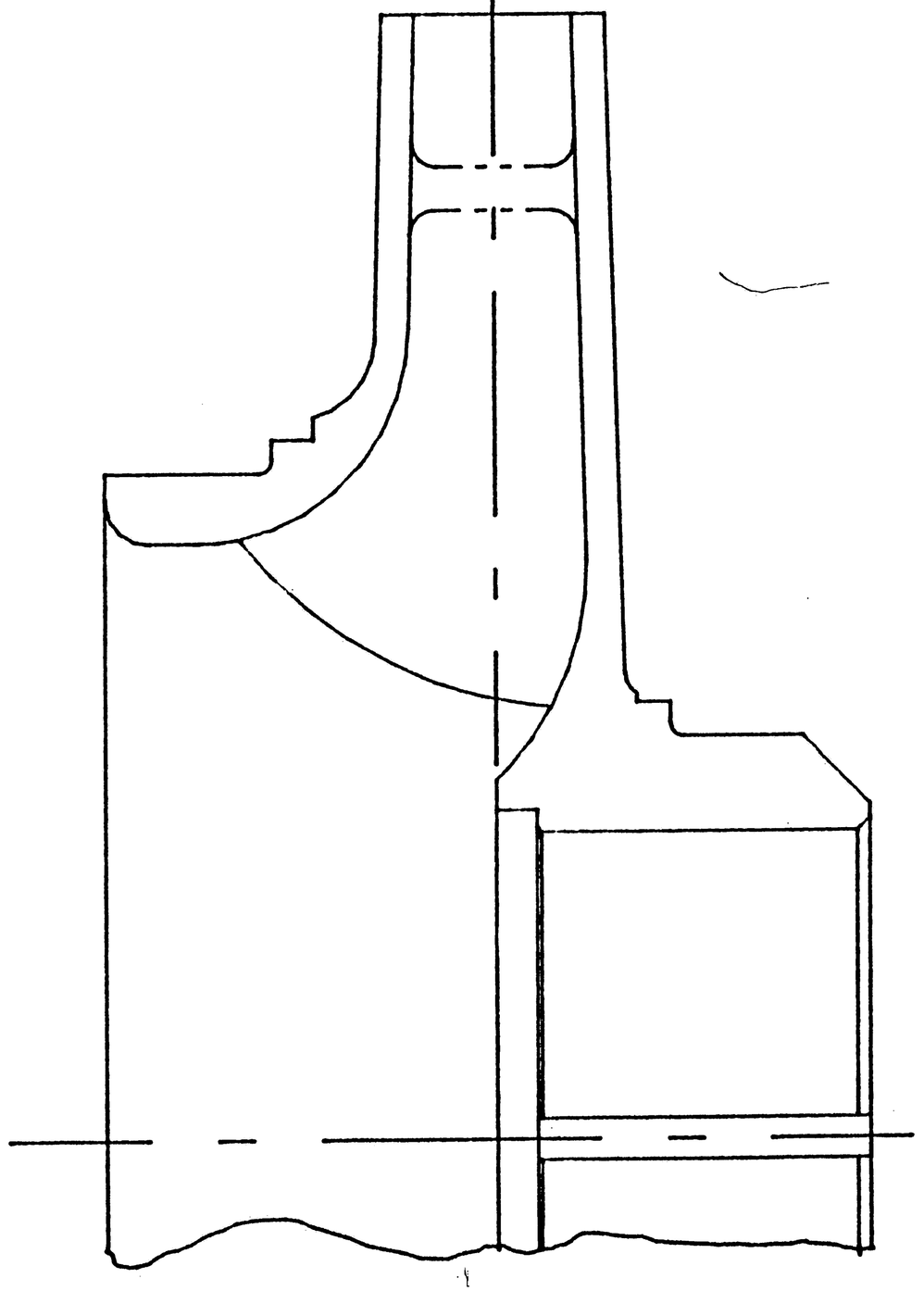


Fig. 6 - Impeller

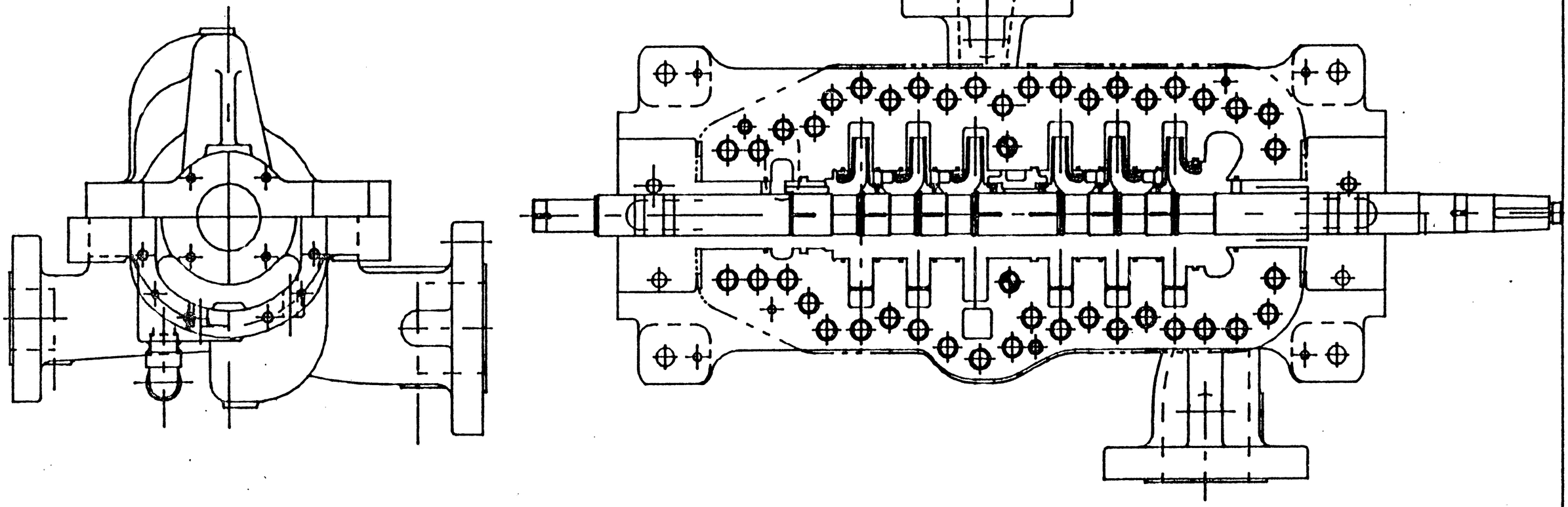


Fig. 7 - Pump Casing

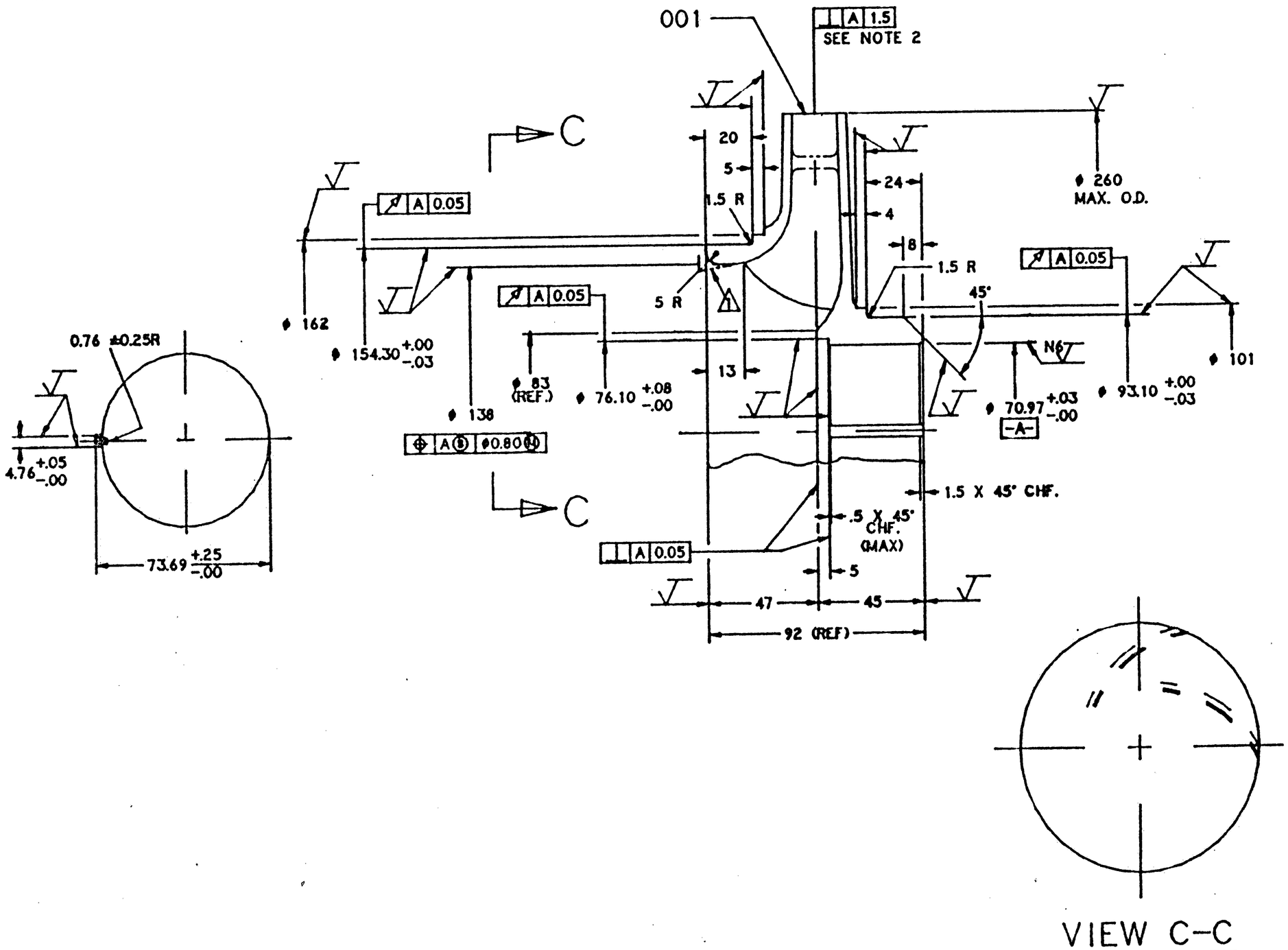


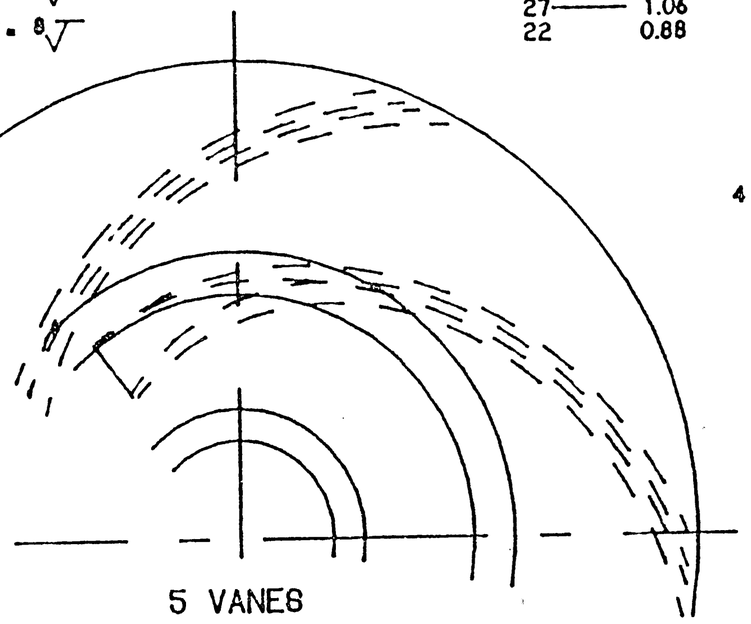
Fig. 8 - Impeller Machining Drawing

METRIC ENGLISH FINISH FINISH

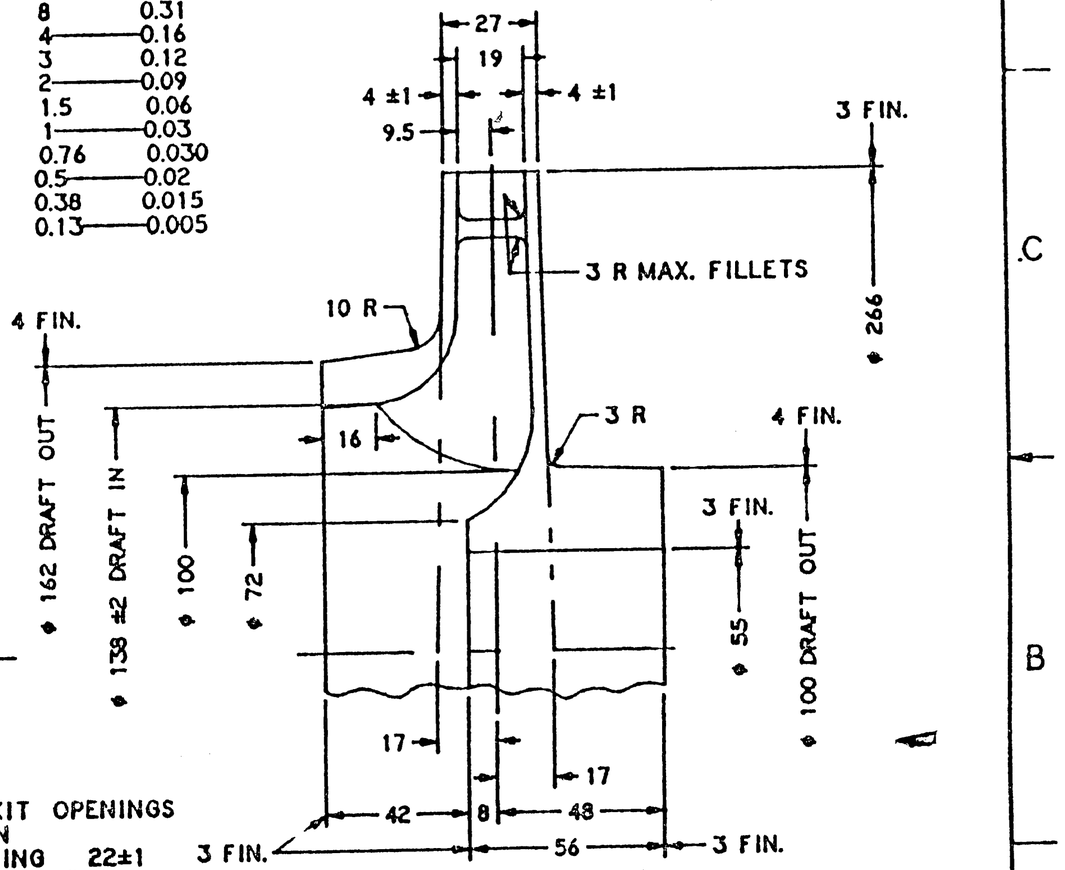
N10 ▽ = 500 ▽
N9 ▽ = 250 ▽
N8 ▽ = 125 ▽
N7 ▽ = 63 ▽
N6 ▽ = 32 ▽
N5 ▽ = 16 ▽
N4 ▽ = 8 ▽

MM	IN
266	10.50
162	6.38
138	5.44
100	3.94
72	2.84
56	2.22
55	2.16
48	1.88
47	1.84
42	1.66
27	1.06
22	0.88

MM	IN
19	0.75
17	0.66
16	0.62
10	0.41
9.5	0.38
8	0.31
4	0.16
3	0.12
2	0.09
1.5	0.06
1	0.03
0.76	0.030
0.5	0.02
0.38	0.015
0.13	0.005



5 VANES



UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS

APPROVALS	DATE
FOUNDRY REP.	
CAMERON I.E. REP.	
CAMERON ENG. REP.	

CHECK INLET AND EXIT OPENINGS ON FINISHED PATTERN
AVERAGE INLET OPENING 22±1
AVERAGE EXIT OPENING 47±1
VANE SPACING TO BE HELD TO ±2
FOR VANE LAYOUT SEE DWG. 3X10DA3ANX0
ALL DIMENSIONS INCLUDE FINISH ALLOWANCE

USED ON 3X10DA "E" HYD.
PRECISION CASTING
-1ST STG.-

PATTERN SUITABLE FOR PRECISION CASTING ONLY

MAXIMUM ACCEPTABLE CAST SURFACE FINISH
125RMS ✓ ALL OVER

IMPELLER L.H. (CAST. DWG.)

62216734 00

CAD

Fig. 9 - Casting Drawing

5.5 Special Cases

The order requirements are often such that the same castings can be used for different orders where the same hydraulic performance characteristics can be applied, but different machining operations for the impellers outside contours are required (i.e. repairs in the field, etc.) In such cases only a new machining drawing is required and the casting is readily available either from the storeroom or can be made according to the existing casting drawing.

6. INDUSTRIAL ENGINEERING

Upon receiving the impeller drawings generated in Engineering, Industrial Engineering determines the tooling that is required to machine the part, designs or adjusts existing fixtures for it and create a NC tape or a DNC file.

There are several parametric computer programs that are currently utilized by the NC designers. These programs are used to generate a NC tool path on impellers, shafts, casings and others. As an example, a portion of a computer program for the NC tool path generation for the DA-line pump impellers is shown on Figure 10. This program contains header portion, a geometry definition portion, a COMPACT II parametric code and the tool selection portion.

The NC designer will extract the required information from the drawing and enter it at the proper place in the file. For example, the numerical value for the impeller O.D. will be entered and program will assign it to a variable.

Next, miscellaneous NC machining data, such as coolant type, speeds, etc., will be appended to the file. The mylar NC tape is then generated or the complete NC file is downloaded to the DNC machine. Industrial Planning and Shop Floor Control will assure the correct machining schedule and the downloading time.

FIGURE 10

NC-FILE LISTING

```
INSRT,#
$\\\NCG*TOOLBUF///
DRAW,FRAME,PT(-5.1631ZA,-.5191XA),PT(7.4323ZA,6.1984XA)
DVR99,                    $ NO. OF RADII ON SHROUD CONTOUR
DVR1,           /25.4 $ MAX O.D
DVR2,           /25.4 $ HUB SIDE RING FIT SHOULDER DIAMETER
DVR3,           /25.4 $ HUB SIDE RING FIT DIA (SPLIT TOLERANCE)
DVR4,           /25.4 $ HUB CHAMFER LENGTH
DVR5,           /25.4 $ BORE DIA (SPLIT TOLERANCE)
DVR6,           /25.4 $ BORE LENGTH HUB FACES
DVR7,           /25.4 $ OVERALL LENGTH
DVR8,           /25.4 $ SUCT. SIDE BORE FACE DIA
DVR9,           /25.4 $ EYE BORE DIA
DVR10,          /25.4 $ SHROUD SIDE RING FIT DIA (SPLIT TOLERANCE)
DVR11,          /25.4 $ SUCTION SIDE RING FIT SHOULDER DIAMETER
DVR13,          /25.4 $ HUB SIDE VANE RUNOUT DIAMETER (CAST PRINT)
DVR14,          /25.4 $ EXIT OPENING (CAST PRINT)
DVR15,          /25.4 $ ZB TO SUCT. SIDE SHROUD WALL AT RADIUS (CAST PRINT)
DVR16,          /25.4 $ ZB TO HUB SIDE SHROUD WALL AT RADIUS (CAST PRINT)
DT)
DVR55,          /25.4 $ HUB TURN STEP DIAMETER
DVR56,          /25.4 $ EYE BORE DIA (CAST PRINT)
DVR57,          /25.4 $ SUCT. HUB FACE TO BORE HUB FACE (CAST PRINT)
DVR60,          /25.4 $ SUCTION SIDE HUB FACE TO VANE TIP
DVR61,                $ TAPER ANGLE IN EYE BORE
DVR63,          /25.4 $ KEYWAY DIMENSION (SPLIT TOLERANCE)
DVR64,                $ HUB CHAMFER ANGLE
DVR65,          /25.4 $ VANE COUNTERBORE DIAMETER
DVR72,                $ MATERIAL SPEC.
$END OF VARIABLES
```

7. ENGINEERING/INDUSTRIAL ENGINEERING INTEGRATION

When the conceptual diagram of Order Processing was completed, many inefficiencies in the operations of the individual departments and interdepartmental communications became apparent.

Among the most obvious inefficiencies were noted:

1. There is a duplication of effort:
   Engineering generates the drawings from a known input data; Industrial Engineering (NC) extracts the same data from the drawings (ref. Figure 3)
2. It takes considerably more time to dimension the geometry on the drawing than it does to generate geometry itself (ref. Figure 4)
3. It takes too much time to check the drawing after it is created.(a bottleneck operation) (ref. Figure 4)
4. There is no existing library of impeller casting and machining files.

The enhanced communications between the departments needed to be established. The decision was made to approach the problem from several directions. The present thesis covers the implementation of the integration strategy between the Engineering and Industrial Engineering Departments.

The writer was given a responsibility to establish a communication channel between the Engineering and Industrial Engineering to solve the first of the above mentioned inefficiencies. This dual communication channel would provide the automatic information transfer from Design Engineering to the Industrial Engineering departmental computers and vice versa.

As a result of this work, by accessing the geometrical information on part files and analyzing it, the parametric information required by the I.E. to create a NC machining or casing file can now be generated at the design stage.

An example of the data extracted from the machining and casting drawings is shown in Appendix B.

The modifications introduced to enhance Engineering functions are shown on Figure 3 in double frames.

8. ENHANCED ENGINEERING FUNCTION

Figure 13 shows a modified procedure in communicating information from the Engineering to Industrial Engineering Departments.

A computer program "NCIMP" for the parametric data extraction from the DA-line pump impeller machining and casting part files has been written. Part file in this context means computer database file. The blank modules were also provided with this initial work for the data extraction for the parts other than impellers in the future. These programs are "NCSHAFT" which will extract information from the shaft part file; "NCDRUM" which will extract data from the balancing drum part file, and others.

The programs are written in IAGL (Interactive Applicon Graphics Language) and AGL (Applicon Graphics Language (compilable). More information on the programming languages and techniques is given in Appendix A.

Program Logic and Operation

The "NCIMP" program is invoked when the designer types in /NC. The program will start execution if the "RELEASE_STATUS"="ARCHIVE", i.e. the drawings have been chceked and approved.

Depending whether the drawing is a machining or a casting, different submodules are executed: "IMPMACH" or "IMPCAST". The program logic flows in a way that it will capture the geometric pattern and the necessary data which varies with the type of the input file. Figure 13 is a picture of a finished machined pump impeller. The program first finds the reference lines to establish a local X-Y coordinate system. It then locates a starting line element using subroutine "PICKL" (line L1, Figure 14) and extracts the endpoints of this line. The subroutines "XENDL","YENDL" rearrange the endpoints in the desired direction of increased coordinates. Next, the pointer moves to the endpoint P2 and searches for an arc using subordinate "PICKC"; extracts endpoints of that arc and

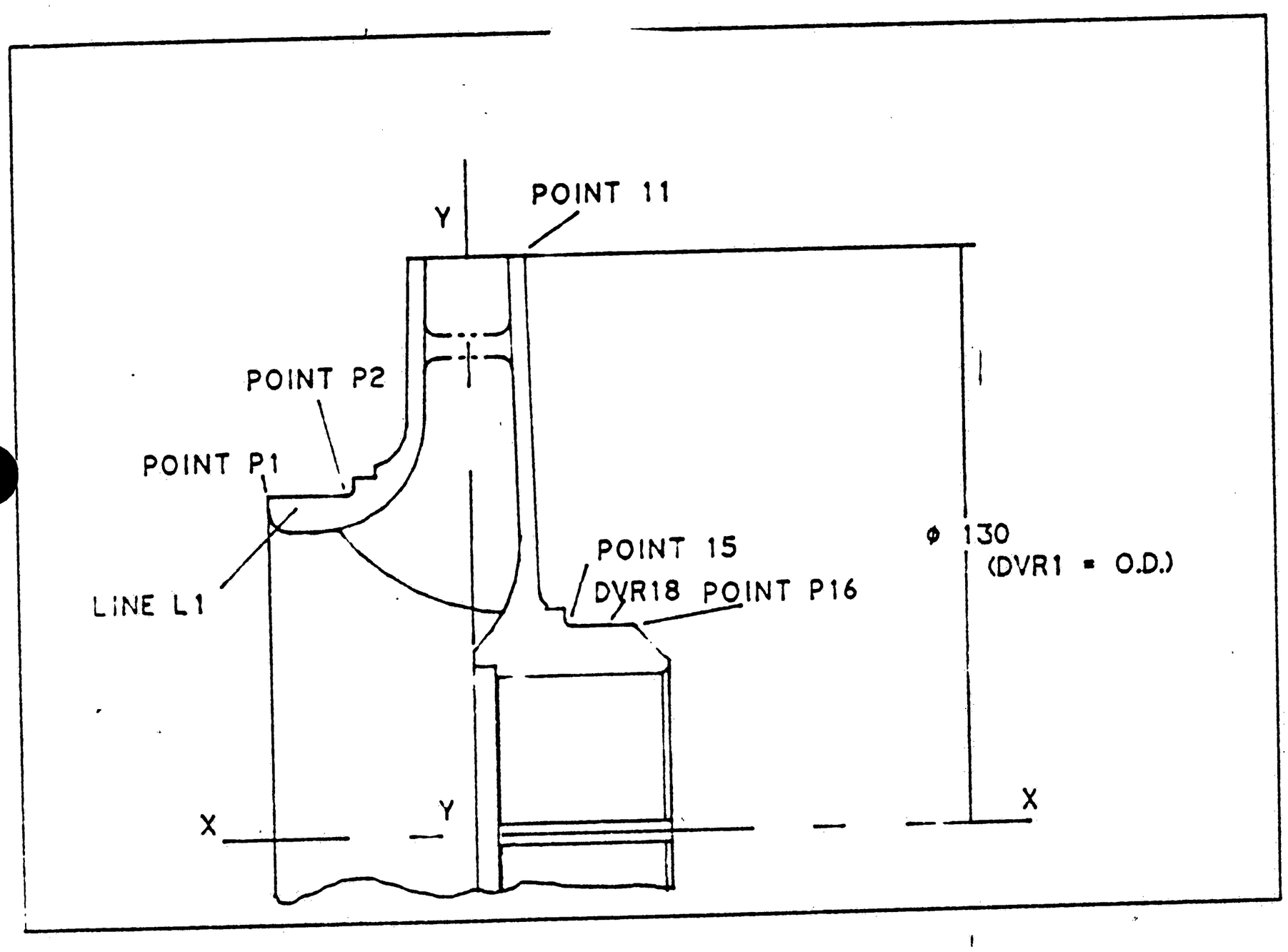


Fig. 14 - Variables, Extracted From Geometry

The execution continues in a similar way until all of the geometry required for the following NC processing is gathered. During this process several subroutines check the validity of the geometric data. For example, subroutine "YSAME" checks if the lines endpoints have the same X-coordinate, if not issues the warning message, thus informing that the line is not horizontal. Similarly, subroutine "XSAME" informs if the line is not vertical.

The subroutines "IMPMACH2" and "IMPCAST2" calculate the values of the DVR-variables needed for the NC machining files. For example, variable DVR1 (Figure 10) describes the impeller outside diameter (O.D.) The program provides the value for this variable by using Y-coordinate of the point 1 multiplied by 2 (Figure 14). Similarly, ring fit length (variable DVR18 Fig. 10) is calculated as a difference in the values of X-coordinates of points P16 and P15. (see Appendix B for subroutines descriptions).

When all of the required DVR-variables receive their values the program writes them out to the external file. In the case of the machining drawing, this file will have an extension specification .MCH.. In the case of the casting drawing this file will be .CAS..(See Appendix B) The files will reside on the Engineering VAX disk under the directory E::FDAZERO :(CADCAM) until another program will reload them to the Manufacturing VAX under the M::FDAZERO:(CADCAM). This is accomplished with the use of the Local Area Network System (LAN). When file transfer is completed and Manufacturing VAX signals the successful files receiving, the files are deleted from the Engineering VAX. The bottom portion of Fig. 13 shows the Network Communication between the VAX'es.

9. ENHANCED NC FUNCTIONS

AND INTERFACE FROM I.E. TO SHOP

As was described in the previous section, two NC files are submitted from the Engineering to Manufacturing mainframe computer. There is one more file required, however, for generation of a complete NC input file. This file is called a Routing file and has a designed extension .RTG . This file is supplied from Planning and Inventory. It contains a description of the material used and other relevant information. A special computer program written in DCL* combines these 3 files together and creates a final NC file. This file can be replayed on the Graphical computer terminal for visual verifications and then downloaded to NC machines in the Machine Shop. This procedure is shown on Figure 15.

*DEC Control Language

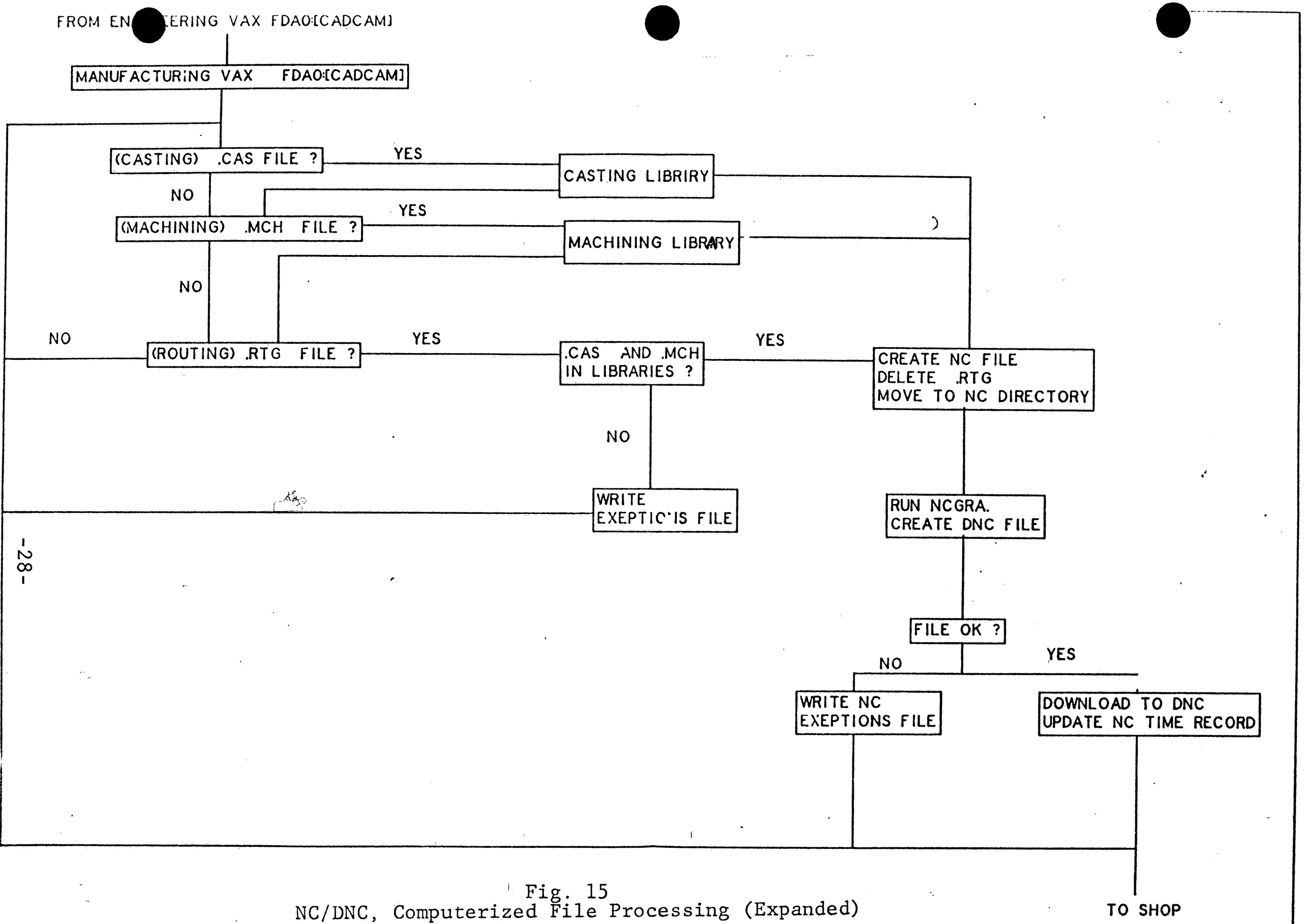


Fig. 15
NC/DNC, Computerized File Processing (Expanded)

19. CONCLUSIONS

The present trends in the Manufacturing Community towards the Factory Automation and utilization of the newest developments in CIM technology find direct and immediate applications at the Ingersoll-Rand Company. As a part of the General Strategic Plan to automate the elements of the manufacturing process, the current work provided significant benefits to the integrated approach in the area of Computer-Aided-Design-and Manufacturing (CAD/CAM).

As a result of this approach towards the integration of engineering and manufacturing functions in the company, the time required to design and manufacture a pump impeller was reduced by at least one order of magnitude which manifested itself in savings in cost and reduction in lead time. At the same time, the quality of the product increased due to the increased consistency and repeatability of the developed procedure.

Due to the modular structure of the developed procedure, new modules can be added as they are developed and updated.

The Order Processing chart is also used as a guidance for the further improvements in the automation of operations.

11. FUTURE PLANS AND RECOMMENDATIONS

As was mentioned earlier, the developed procedure does not represent a "static structure". On the contrary, it is a dynamic evolving process. In the Section 6 several areas of potential improvements were listed. In that respect, the following has been planned for the future:

1. Develop more modules to process parts other than pump impellers. These will include shafts, balancing drums, rings, etc. In fact, the writer is beginning to work on a completely computerized parametric procedure of the design, dimensioning and generating the NC input files from their geometry..
2. The currently existing computer program used by the other departments (Financial Control, Planning, etc.) will be analyzed and integrated with the Engineering and Manufacturing operations. In particular, currently available automated programs for the generation of the Bill of Materials will be enhanced to provide a link to Engineering.
3. Inventory Control and Forecasting will be combined with the other Engineering functions.

APPENDICES

APPENDICES

APPENDIX A

IAGL, AGL, DBA PROGRAMMING*

*See "Applicon" programming manuals for more information.

In this apprendix, a brief description of IAGL, AGL and DBA is given. Since it is not a purpose of the current thesis work, to educate the reader on using "Applicon" programming language, the writer will suggest for those who might be interested in more detailed description, refer to the "Applicon" programming manuals.

VAX-11 family of machines support several high-level languages which include FORTRAN, VAX Macro (assembler), PL/I AGL (VAX-11 PL/I with graphics extensions), and others.

IAGL resembles AGL very closely. In fact,it is a subset of the full AGL language, but there are a few differences. Being interpretive (somewhat similar to BASIC in that respect), IAGL represents a good choice for a computer programmer at the initial stage, i.e. when the program is first written and being debugged.

When all development work or writing an IAGL program is completed, it may be desirable to correct it (or modify) to AGL language. AGL language is compilable, and therefore is considerably faster than IAGL in execution.

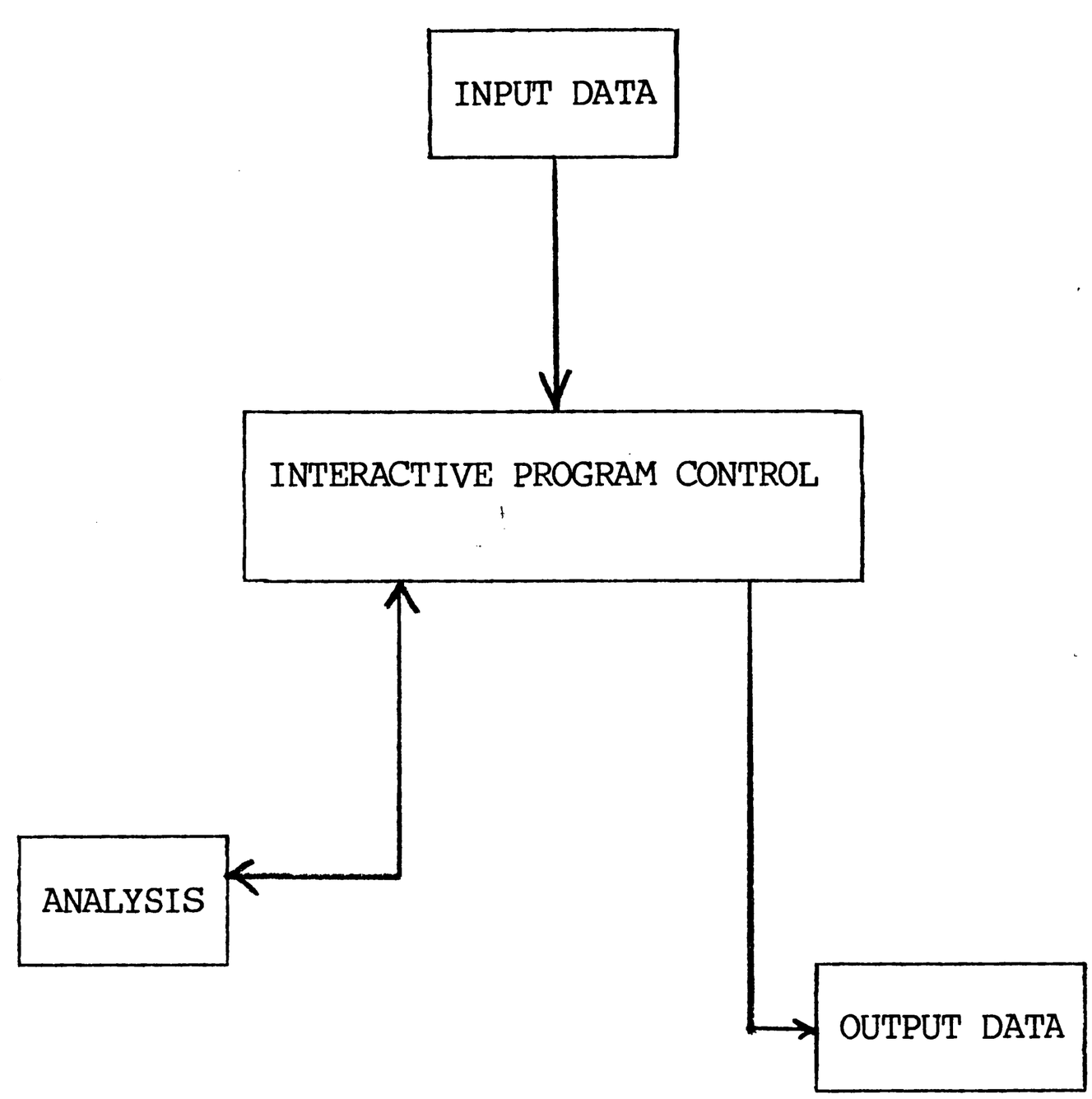
In addition to more "conventional" programming techniques,"Applicon" presents a powerful set of so called DBA routines. The main purpose and advantage in using these is very fast, and flexible direct access to the database files by-passing the usual "Editor" graphics interpretor routines.

The speed of access becomes very much faster than using graphics interpretor. This approach becomes unsurpassable when dealing with large databases and numerous geometry manipulations.

Appendix B (IAGL)

NC output files, program logic descriptions and listing of program portions.

B.1 Program Logic Diagram



Input Data - provided by the designer to describe the geometry

Analysis - involves routines performing various calculations and geometry extractions

Output Data - writes the output file for the NC processing

Interactive Program Control - links various routines and oversees general operations.

B.2 Program Routines Grouping

B.2.1 Routines for Input Data Handling

"NC", "NCIMP", "IMPROUT", "NCSHAFT", 'NCDRUM"

B.2.2 Routines for Interactive Program Control Handling

"IMPMACHI", "IMPMACH2", IMPCAST1", "IMPCAST2"

B.2.3 Routines for Analysis Handling

"XENDL", "YENDL", "XENDCIR", "YENDCIR", "PICKL", "PICKC",

"XSAME", "YSAME", "DELLI", "ANGCHECK",

B.2.4 Routines for Output Data Handling

"CHARACTER", "FWRITE"

B.2.1.1 "NC"

Initial entry. Presents a choice of currently available modules and executes the selected one.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]NC.CPR;10

NC:CPROC(%NCPROGRAM PROMPT('WHICH NC PROGRAM DO YOU WANT TO RUN ?')
        CHOICES(
        ('NCIMP','GENERATE IMPELLER NC-OUTPUT'),
        ('NCSHAFT','GENERATE SHAFT NC-OUTPUT'),
        ('NCDRUM','GENERATE DRUM NC-OUTPUT'),
('NCSLEEVE','GENERATE SLEEVE NC-OUTPUT'),
('GUESS_WHAT?','NOT IMPLEMENTED')));
DCL RPART CHAR(15) VAR;
RPART='%NCPROGRAM';
\%NCPROGRAM
%END
```

B.2.1.2 "NCIMP"

Starts the series of routines to access the impeller geometry and extract information required for the NC file.

Invokes the routines to operate either on the impeller machining or casting database.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]NCIMP.CPR;14

NCIMP:CPROC(%INCH_MET PROMPT('WHAT UNITS ?')
        CHOICES(
        'INCHES','METRIC'),
%PROGRAM PROMPT('SELECT THE IMPELLER TYPE DRWG.')
        CHOICES(
        ('IMPCAST1','IMPELLER CASTING DRWG'),
        ('IMPMACH1','IMPELLER MACHINING DRWG')),
        %CASTDWGNO PROMPT('ENTER CASTING DRAWING NUMBER'));
DCL T CHAR(1);
DCL (RMA_CA,INCH_MET,CASTDWGNO) CHAR(15) VAR;
INCH_MET='%INCH_MET';
RMA_CA='%PROGRAM';
CASTDWGNO='%CASTDWGNO';
IF RMA_CA = 'IMPCAST1' THEN DO;
MESS 'YOU MUST HAVE CASTING DRWG. ON THE SCREEN. DO YOU ? Y/N';
GETV TEXT 'T' STRING;
IF T ^= 'Y' THEN ABORT ;
END;
IF RMA_CA = 'IMPMACH1' THEN DO;
MESS  'YOU MUST HAVE MACHINING DRWG. ON THE SCREEN. DO YOU ? Y/N' ;
GETV TEXT 'T' STRING;
IF T^= 'Y' THEN ABORT ;
END;
\%PROGRAM
%END
```

B2.1.3 "IMPROUT"

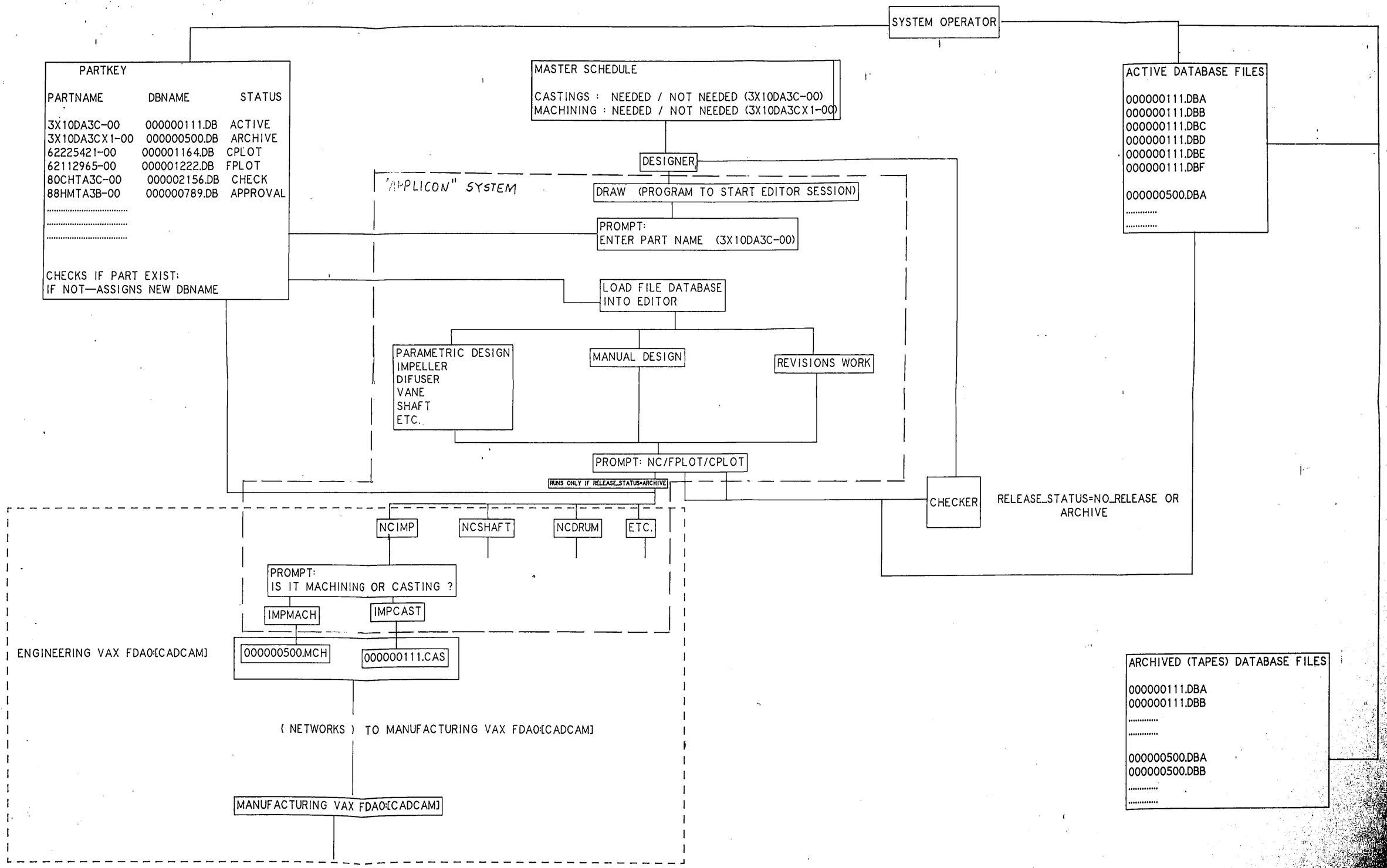
Supplies additional external information not available from the graphical database.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]IMPROUT.CPR;15

IMPROUT:CPROC(
        %DVR28 PROMPT('ENTER SHROUD FINISH RING O.D.'),
        %DVR29 PROMPT('ENTER HUB FINISH RING O.D.'),
        %SUTTON PROMPT('IS SUTTON OPERATION NEEDED ? Y/N'),
        %DVR63 PROMPT('ENTER KEYWAY HEIGHT'),
        %KEYWIDTH PROMPT('ENTER KEYWAY WIDTH'),
        %DVR65 PROMPT('ENTER VANE COUNTERBORE DIA.'));
DCL (DVR28,DVR29,DVR48,DVR63,DVR65) FLOAT;
DCL (SUTTON,KEYWIDTH) CHARACTER(20) VAR;
/*****************************/;
DVR28=%DVR28;
DVR29=%DVR29;
SUTTON='%SUTTON';
DVR48=0; IF SUTTON='Y' THEN DVR48=-1.;
DVR63=%DVR63;
KEYWIDTH='%KEYWIDTH';
DVR65=%DVR65;
DFL_2=DVR28; \CHARACTER ; CH32_6='DVR28,'!!CH32_6; \FWRITE
DFL_2=DVR29; \CHARACTER ; CH32_6='DVR29,'!!CH32_6; \FWRITE
DFL_2=DVR48; \CHARACTER ; CH32_6='DVR48,'!!CH32_6; \FWRITE
CH32_6='%SUTTON'; CH32_6='$ SUTTON: '!!SUTTON; \FWRITE
CH32_6='%KEYWIDTH'; CH32_6='$ KEYWIDHT= '!!KEYWIDTH; \FWRITE
DFL_2=DVR63; \CHARACTER ; CH32_6='DVR63,'!!CH32_6; \FWRITE
DFL_2=DVR65; \CHARACTER ; CH32_6='DVR65,'!!CH32_6; \FWRITE
%END
```

FIG. 13

ENGINEERING DESIGN.
CAE : CAD + CAM
(ENHANCED WITH NC-LINK)



B.2.2.1 "IMPMACH1"

Starts extracting information about the selected geometry components in the parametrically predefined order.

Invokes a series of the auxiliary routines.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]IMPMACH1.CPR;21

IMPMACH1:CPROC;
DCL (LHCL,LVCL) LINE 3D;
DCL (L,LT,L1,L2,L3,L4,L5,L6,L7,L8,L9,L10) LINE 3D;
DCL (L11,L12,L13,L14,L15,L16,L17,L18,L19,L20) LINE 3D;
DCL (L21,L22,L23,L24,L25,L26,L27,L28,L29,L30) LINE 3D;
DCL (P,PS,PL,PPS,PPL,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10) POINT 3D;
DCL (P11,P12,P13,P14,P15,P16,P17,P18,P19,P20) POINT 3D;
DCL (P21,P22,P23,P24,P25,P26,P27,P28,P29,P30) POINT 3D;
DCL (C,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10) CIRCLE 3D;
DCL PARPEN FIXED;
DCL (A,R5) FLOAT;
DCL (XBASE,YBASE) FLOAT;
SPEC DRAW PERL ON;
SPEC WORK MSGA 10;
SPEC WORK EXPAN ON;
SPEC WORK PICKL 8;
MESS 'DIGITIZE HORIZONTAL C-LINE ';
GETV LINE 'LHCL' PICK NEAR ONE MAIN;
\XENDL
YBASE=YCRD(PL);
MESS 'DIGITIZE VERTICAL C-LINE';
GETV LINE 'LVCL' PICK NEAR ONE MAIN;
\YENDL
XBASE=XCRD(PL);
MESS 'DIGITIZE THE STARTING LINE';
GETV LINE 'L1' PICK NEAR ONE MAIN;
L=L1;
*************************************/;
```

```
/*********************************************/
PARPEN=1;
\XENDL
P1=PS;
P2=PL;
MODI DEL PICK T1I;
SPEC WORK PICKL 2;
P=P2;
\PICKC
C1=C;
\XENDCIR
P3=PL;
P=P2;
\DELLI
P=P3;
\PICKL
L2=L;
\YENDL
P4=PL;
P=P3;
\DELLI
P=P4;
\PICKL
L3=L;
PARPEN=1;
\XENDL
P5=PL;
P=P4;
\DELLI
P=P5;
\PICKL
L4=L;
\YENDL
P6=PL;
P=P5;
\DELLI
P=P6;
\PICKC
C2=C;
\YENDCIR
P7=PL;
P=P6;
\DELLI
P=P7;
\PICKL
L5=L;
\YENDL
P8=PL;
P=P7;
\DELLI
P=P8;
\PICKL
L6=L;
PARPEN=1;
\XENDL
P9=PL;
P=P8;
\DELLI
P=P9;
\PICKL
\ANGCHECK
```

```
IF A>10 THEN DO;
LT=L;
\DELLI
END;
IF A>10 THEN \PICKL
L7=L;
PARPEN=1;
\XENDL
P10=PL;
P=P9;
\DELLI
IF A>10 THEN ADD LINE (LT);
P=P10;
\PICKL
\ANGCHECK
IF A>10 THEN DO;
LT=L;
\DELLI
END;
IF A>10 THEN \PICKL
L8=L;
PARPEN=1;
\XENDL
P11=PL;
P=P10;
\DELLI
IF A>10 THEN ADD LINE (LT);
P=P11;
\PICKL
L9=L;
\YENDL
P12=PS;
P=P11;
\DELLI
P=P12;
\PICKC
C3=C;
\YENDCIR
P13=PS;
P=P12;
\DELLI
P=P13;
\PICKL
L10=L;
\YENDL
P14=PS;
P=P13;
\DELLI
P=P14;
\PICKL
L11=L;
PARPEN=1;
\XENDL
P15=PL;
P=P14;
\DELLI
P=P15;
\PICKL
L12=L;
\YENDL
```

```
P16=PS;
P=P15;
\DELLI
P=P16;
\PICKC
C4=C;
\XENDCIR
P17=PL;
P=P16;
\DELLI
P=P17;
\PICKL
L13=L;
PARPEN=1;
\XENDL
P18=PL;
P=P17;
\DELLI
P=P18;
\PICKL
L14=L;
\XENDL
P19=PL;
P=P18;
\DELLI
P=P19;
\PICKL
L15=L;
\YENDL
P20=PS;
P=P19;
\DELLI
P=P20;
\PICKL
\ANGCHECK
IF A>80 THEN DO;
LT=L;
\DELLI
END;
IF A>80 THEN \PICKL
L16=L;
\XENDL
P21=PS;
P=P20;
\DELLI
IF A>80 THEN ADD LINE (LT);
P=P21;
\PICKL
\ANGCHECK
IF A>10 THEN DO;
LT=L;
\DELLI
END;
IF A>10 THEN \PICKL
L17=L;
PARPEN=1;
\XENDL
P22=PS;
P=P21;
\DELLI
```

```
IF A>10 THEN ADD LINE (LT);
P=P22;
\PICKL
\ANGCHECK
IF A>80 THEN DO;
LT=L;
\DELLI
END;
IF A>80 THEN \PICKL
L18=L;
\XENDL
P23=PS;
P=P22;
\DELLI
IF A>80 THEN ADD LINE (LT);
P=P23;
\PICKL
LT=L;
\YENDL
PPL=PL;
PPS=PS;
P=P23;
\DELLI
\PICKL
\YENDL
IF YCRD(PL)>YCRD(PPL) THEN L19=L;
IF YCRD(PL)<YCRD(PPL) THEN L19=LT;
IF YCRD(PL)>YCRD(PPL) THEN P24=PL;
IF YCRD(PL)<YCRD(PPL) THEN P24=PPL;
P=P23;
IF YCRD(PL)>YCRD(PPL) THEN \DELLI
IF YCRD(PL)>YCRD(PPL) THEN ADD LINE (LT);
P=P24;
\PICKL
L20=L;
PARPEN=1;
\XENDL
P25=PS;
P=P24;
\DELLI
P=P25;
\PICKL
LT=L;
\YENDL
PPL=PL;
PPS=PS;
P=P25;
\DELLI
\PICKL
\YENDL
IF YCRD(PL)>YCRD(PPL) THEN L21=L;
IF YCRD(PL)<YCRD(PPL) THEN L21=LT;
IF YCRD(PL)>YCRD(PPL) THEN P26=PL;
IF YCRD(PL)<YCRD(PPL) THEN P26=PPL;
P=P25;
IF YCRD(PL)>YCRD(PPL) THEN \DELLI
IF YCRD(PL)>YCRD(PPL) THEN ADD LINE (LT);
```

```
P=P1;
\PICKL
L22=L;
\YENDL
P27=PS;
P=P1;
\DELLI
P=P27;
\PICKC
C5=C;
\XENDCIR
P28=PL;
\PICKL
L23=L;
\XENDL
P29=PL;
P=P29;
\PICKC
C6=C;
\XENDCIR
P30=PL;
/* ADDING DELETED LINES AND ARCS *****/
ADD LINE (L1) (L2) (L3) (L4) (L5) (L6) (L7) (L8) (L9) (L10)
(L11) (L12) (L13) (L14) (L15) (L16) (L17) (L18) (L19) (L20)
(L21) (L22);
ADD CIRCLE (C1) (C2) (C3) (C4) ;
/********* GEOMETRY EXTRUCTION ***/
R5=RADIUS(C5);
\IMPMACH2
SPEC WORK PICKL 8;
%END
```

B.2.2.2 "IMPMACH2"

Calculates values for DVR-variables needed for the NC input files.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]IMPMACH2.CPR;25

IMPMACH2:CPROC;
DCL (DVR1,DVR2,DVR3,DVR4,DVR5,DVR6,DVR7,DVR8,
DVR9,DVR10,DVR11,DVR14,DVR18,DVR188,DVR23,
DVR24,DVR31,DVR32,DVR44,DVR45,DVR47,DVR55,
DVR60,DVR61,DVR64) FLOAT;
DCL (DX,DY) FLOAT;
DCL (R1,R2,R3,R4,R5A,R6) CHAR(50) VAR;
DCL (PARTNO,DELPARTNO) CHAR(50) VAR;
PARTNO='FDA0:[CADCAM]'!!STOREDB!!'.MCH';
/** IF JOHN LIED : PARTNO='FDA0:[ENG.NELIK.NC]'!!STOREDB!!'.MCH';**/;
R1='$ '!!RPART;
R2='$ '!!RMA_CA;
R3='$ CAST $ '!!CASTDWGNO;
R4='$ '!!STOREDB;
R5A='$ '!!DRG_1;
R6='$ '!!INCH_MET;
/*****************************************/
```

```
/* DVR99 ALWAYS 1 RADIUS */
DVR1=(YCRD(P8)-YBASE)*2; /**ALWAYS MAX O.D. **/;
DVR2=(YCRD(P15)-YBASE)*2;
DVR3=(YCRD(P18)-YBASE)*2;
DVR4=XCRD(P19)-XCRD(P18);
DVR5=(YCRD(P21)-YBASE)*2;
DVR6=XCRD(P20)-XCRD(P25);
DVR7=XCRD(P20)-XCRD(P1);
DVR8=(YCRD(P26)-YBASE);
DVR9=(YCRD(P29)-YBASE)*2;
DVR10=(YCRD(P1)-YBASE)*2;
DVR11=(YCRD(P4)-YBASE)*2;
DVR14=XCRD(P10)-XCRD(P9);
/*DVR15 FROM CASTING */
/*DVR16 FROM CASTING */
DVR18=XCRD(P19)-XCRD(P15);
DVR188=XCRD(P4)-XCRD(P1);
/* DVR21 FROM CASTING */
DVR23=XCRD(P19)-XCRD(P13);
DVR24=R5;
/* DVR26 FROM CASTING */
/* DVR28 FROM RINGS */
/* DVR29 FROM RINGS */
/* DVR30 FROM ROUTING */
DVR31=XCRD(P19)-XCRD(P25);
DVR32=XCRD(P20)-XCRD(P21);
DVR44=XCRD(P24)-XCRD(P25);
DVR45=XCRD(P5)-XCRD(P1);
DVR47=(YCRD(P25)-YBASE)*2;
/* DVR48 FROM ROUTING  */
/* DVR49 DIFFERENCE IN DESIGN ! */
/* DVR50 FROM CASTING */
/* DVR51 FROM CASTING */
/* DVR53 FROM CASTING */
/* DVR54  FROM CASTING */
DVR55=(YCRD(P19)-YBASE)*2;
/* DVR57 FROM CASTING */
DVR60=XCRD(P30)-XCRD(P1);
DY=YCRD(P28)-YCRD(P29);
DX=XCRD(P29)-XCRD(P28);
DVR61=ATAND(DY/DX);
/* DVR63 FROM KEYWAY */
DY=YCRD(P18)-YCRD(P19);
DX=XCRD(P19)-XCRD(P18);
DVR64=ATAND(DY/DX);
/* DVR65 ENTERED .......*/
/* DVR72 ENTERED .......*/
/*************************************************/
```

```
VMS 'DELETE FDA0:[CADCAM]00000*.MCH;*' NONE;
CLOSE FILE(FILE1);
OPEN FILE(FILE1) TITLE(PARTNO) OUTPUT RECORD;
WRITE FILE(FILE1) FROM (R1);
WRITE FILE(FILE1) FROM (R2);
WRITE FILE(FILE1) FROM (R3);
WRITE FILE(FILE1) FROM (R4);
WRITE FILE(FILE1) FROM (R5A);
WRITE FILE(FILE1) FROM (R6);
DFL_2=DVR2; \CHARACTER ; CH32_6='DVR2,'!!CH32_6; \FWRITE
DFL_2=DVR3; \CHARACTER ; CH32_6='DVR3,'!!CH32_6; \FWRITE
DFL_2=DVR4; \CHARACTER ; CH32_6='DVR4,'!!CH32_6; \FWRITE
DFL_2=DVR5; \CHARACTER ; CH32_6='DVR5,'!!CH32_6; \FWRITE
DFL_2=DVR6; \CHARACTER ; CH32_6='DVR6,'!!CH32_6; \FWRITE
DFL_2=DVR7; \CHARACTER ; CH32_6='DVR7,'!!CH32_6; \FWRITE
DFL_2=DVR8; \CHARACTER ; CH32_6='DVR8,'!!CH32_6; \FWRITE
DFL_2=DVR9; \CHARACTER ; CH32_6='DVR9,'!!CH32_6; \FWRITE
DFL_2=DVR10; \CHARACTER ; CH32_6='DVR10,'!!CH32_6; \FWRITE
DFL_2=DVR11; \CHARACTER ; CH32_6='DVR11,'!!CH32_6; \FWRITE
DFL_2=DVR14; \CHARACTER ; CH32_6='DVR14,'!!CH32_6; \FWRITE
DFL_2=DVR18; \CHARACTER ; CH32_6='DVR18,'!!CH32_6; \FWRITE
DFL_2=DVR188; \CHARACTER ; CH32_6='DVR188,'!!CH32_6; \FWRITE
DFL_2=DVR23; \CHARACTER ; CH32_6='DVR23,'!!CH32_6; \FWRITE
DFL_2=DVR24; \CHARACTER ; CH32_6='DVR24,'!!CH32_6; \FWRITE
DFL_2=DVR31; \CHARACTER ; CH32_6='DVR31,'!!CH32_6; \FWRITE
DFL_2=DVR32; \CHARACTER ; CH32_6='DVR32,'!!CH32_6; \FWRITE
DFL_2=DVR44; \CHARACTER ; CH32_6='DVR44,'!!CH32_6; \FWRITE
DFL_2=DVR45; \CHARACTER ; CH32_6='DVR45,'!!CH32_6; \FWRITE
DFL_2=DVR47; \CHARACTER ; CH32_6='DVR47,'!!CH32_6; \FWRITE
DFL_2=DVR55; \CHARACTER ; CH32_6='DVR55,'!!CH32_6; \FWRITE
DFL_2=DVR60; \CHARACTER ; CH32_6='DVR60,'!!CH32_6; \FWRITE
DFL_2=DVR61; \CHARACTER ; CH32_6='DVR61,'!!CH32_6; \FWRITE
DFL_2=DVR64; \CHARACTER ; CH32_6='DVR64,'!!CH32_6; \FWRITE
/***** WRITING INFO FROM ROUTING ***/;
CH32_6='$ MANUAL INPUT AT CAD'; \FWRITE ;
\IMPROUT ;
CLOSE FILE(FILE1);
%END
```

B.2.2.3 "IMPCAST1"

Similar to "IMPMACH1" except that it is used for the casting drawings.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]IMPCAST1.CPR;13

IMPCAST1:CPROC;
DCL (P,CPC1,PCLINE,PS,PL,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10) POINT 3D;
DCL (P11,P12,P13,P14,P15,P16,P17,P18,P19,P20) POINT 3D;
DCL (L,L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15) LINE 3D;
DCL (C,C1,C2,C3) CIRCLE 3D;
DCL (A,A2,A4) FLOAT;
SPEC DRAW PERL ON;
SPEC WORK MSGA 10;
SPEC WORK EXPAN ON;
SPEC WORK PICKL 8;
MESS ' DIGITIZE THE STARTING LINE';
GETV LINE 'L1' PICK NEAR ONE MAIN;
L=L1;
\XENDL
P1=PS;
P2=PL;
MODI DEL PICK T1I;
SPEC WORK PICKL 2;
P=P2;
\PICKC
C1=C;
\XENDCIR
P3=PL;
P=P2;
\DELLI
P=P3;
\PICKL
L2=L;
\YENDL
P4=PL;
P=P3;
\DELLI
P=P4;
\PICKL
L3=L;
\XENDL
P5=PL;
P=P4;
\DELLI
P=P5;
\PICKL
L4=L;
\YENDL
P6=PS;
P=P5;
\DELLI
```

```
P=P6;
\PICKC
C2=C;
\XENDCIR
P7=PL;
P=P6;
\DELLI
P=P7;
\PICKL
L5=L;
\XENDL
P8=PL;
P=P7;
\DELLI
P=P8;
\PICKL
L6=L;
\YENDL
P9=PS;
P=P8;
\DELLI
P=P9;
\PICKL
L7=L;
\XENDL
P10=PS;
P=P9;
\DELLI
P=P10;
\PICKL
L8=L;
\YENDL
P11=PL;
P=P10;
\DELLI
P=P1;
\PICKL
L9=L;
\YENDL
P12=PS;
P=P1;
\DELLI
P=P12;
\PICKL
L10=L;
\XENDL
P13=PL;
/**** GEOMETRY EXTRACTS **/
CPC1=CENTER(C1);
PCLINE=(P4+P5)/2;
P=P5;
L=L4;
\ANGCHECK
A4=A;
P=P4;
L=L2;
\ANGCHECK
A2=A;
/*** ADDING DELETED LINES AND CIRCLES***/
ADD LINE (L1) (L2) (L3) (L4) (L5) (L6) (L7) (L8) (L9) ;
ADD CIRCLE (C1) (C2) ;
\IMPCAST2
%END
```

B.2.2.4 "IMPCAST2"

Similar to "IMPMACH2" except that it is used for the casting drawings.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]IMPCAST2.CPR;20

IMPCAST2:CPROC;
DCL (DVR15,DVR16,DVR26,DVR50,DVR51,DVR53,DVR54,DVR57) FLOAT;
DCL (DX,DY,XBASE) FLOAT;
DCL (R1,R2,R3,R4,R5A) CHAR(20) VAR;
DCL PARTNO CHAR(50) VAR;
PARTNO='FDA0:[CADCAM]'!!STOREDB!!'.CAS';
/** IF JOHN LIED : PARTNO='FDA0:[ENG.NELIK.NC]'!!STOREDB!!'.CAS';**/;
R1='$ '!!RPART;
R2='$ '!!RMA_CA;
R3='$ ';
R4='$ '!!STOREDB;
R5A='$ '!!DRG_1;
R6='$ '!!INCH_MET;
/**********************************/
XBASE=XCRD(P1);
DX=YCRD(P4)*TAND(90-A2);
DVR15=XCRD(PCLINE)-(XCRD(P4)-DX);
DX=YCRD(P5)*TAND(90-A4);
DVR16=(XCRD(P5)+DX)-XCRD(PCLINE);
DVR26=YCRD(CPC1);
DVR50=YCRD(P5)*2;
DVR51=XCRD(P5)-XCRD(P4);
DVR53=XCRD(P9)-XCRD(PCLINE);
DVR54=XCRD(PCLINE)-XCRD(P10);
DVR57=XCRD(P10)-XCRD(P1);
/*****/;
VMS 'DELETE FDA0:[CADCAM]00000*.CAS;*' NONE;
CLOSE FILE(FILE1);
OPEN FILE(FILE1) TITLE(PARTNO) OUTPUT RECORD;
WRITE FILE(FILE1) FROM(R1);
WRITE FILE(FILE1) FROM(R2);
WRITE FILE(FILE1) FROM(R3);
WRITE FILE(FILE1) FROM(R4);
WRITE FILE(FILE1) FROM(R5A);
WRITE FILE(FILE1) FROM(R6);
DFL_2=DVR15;
\CHARACTER
CH32_6='DVR15,'!!CH32_6;
\FURITE
```

```
\FWRITE
DFL_2=DVR16;
\CHARACTER
CH32_6='DVR16,'!!CH32_6;
\FWRITE
DFL_2=DVR26;
\CHARACTER
CH32_6='DVR26,'!!CH32_6;
\FWRITE
DFL_2=DVR50;
\CHARACTER
CH32_6='DVR50,'!!CH32_6;
\FWRITE
DFL_2=DVR51;
\CHARACTER
CH32_6='DVR51,'!!CH32_6;
\FWRITE
DFL_2=DVR53;
\CHARACTER
CH32_6='DVR53,'!!CH32_6;
\FWRITE
DFL_2=DVR54;
\CHARACTER
CH32_6='DVR54,'!!CH32_6;
\FWRITE
DFL_2=DVR57;
\CHARACTER
CH32_6='DVR57,'!!CH32_6;
\FWRITE
CLOSE FILE(FILE1);
/**************************************/
%END
```

B.2.3.1, B2.3.2, B2.3.3, B2.3.4 "XENDL", "YENDL", "XENDCIR", YENDCIR".

Routines to determine two endpoints of the specific components (lines and arcs) and to rearrange the endpoints in the predefined order.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]XENDL.CPR;3

XENDL:CPROC;
PS=ENDPOINT1(L);
PL=ENDPOINT2(L);
IF XCRD(PL)<XCRD(PS) THEN DO;
P=PS;
PS=PL;
PL=P;
END;
\YSAME
%END
```

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]YENDL.CPR;2

YENDL:CPROC;
PS=ENDPOINT1(L);
PL=ENDPOINT2(L);
IF YCRD(PL)<YCRD(PS) THEN DO;
P=PS;
PS=PL;
PL=P;
END;
\XSAME
%END
```

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]XENDCIR.CPR;2

XENDCIR:CPROC;
PS=ENDPOINT1(C);
PL=ENDPOINT2(C);
IF XCRD(PL)<XCRD(PS) THEN DO;
P=PS;
PS=PL;
PL=P;
END;
%END
```

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]YENDCIR.CPR;2

YENDCIR:CPROC;
PS=ENDPOINT1(C);
PL=ENDPOINT2(C);
IF YCRD(PL)<YCRD(PS) THEN DO;
P=PS;
PS=PL;
PL=P;
END;
%END
```

B2.3.5, B2.3.6 "PICKC","PICKL"

Retrieves a line or arc in the immediate surrounding of the graphical pointer.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]PICKL.CPR;1

PICKL:CPROC;
GETV LINE 'L' PICK NEAR ONE MAIN (P);
%END
```

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]PICKC.CPR;1

PICKC:CPROC;
GETV CIRCLE 'C' PICK NEAR ONE MAIN (P);
%END
```

B.2.3.7, B.2.3.8 "XSAME", "YSAME"

Determines if the line is horizontal or vertical with respect to the local coordinate system.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]XSAME.CPR;1

XSAME:CPROC;
DCL PM POINT 3D;
PM=(PS+PL)/2;
IF XCRD(PL) ^= XCRD(PS)
 THEN IF PARPEN=1 THEN DO;
MESS 'LINE IS NOT VERTICAL ! FIX IT AND RERUN.';
LIST SELECT PICK NEAR ONE MAIN (PM);
SPEC WORK PICKL 8;
ABORT;
END;
 ELSE;
PARPEN=0;
%END
```

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]YSAME.CPR;1

YSAME:CPROC;
DCL PM POINT 3D;
PM=(PS+PL)/2;
IF YCRD(PL) ^= YCRD(PS)
 THEN IF PARPEN=1 THEN DO;
MESS 'LINE IS NOT PARALLEL ! FIX IT AND RERUN.';
LIST SELECT PICK NEAR ONE MAIN (PM);
SPEC WORK PICKL 8;
ABORT;
END;
 ELSE;
PARPEN=0;
%END
```

B.2.3.9 "DELLI"

Deletes the component

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]DELLI.CPR;1

DELLI:CPROC;
MODI DEL PICK NEAR ONE MAIN (P);
%END
```

B.2.3.10 "ANGCHECK"

Determines the direction in which to proceed to retrieve the geometry in the cases of multiple geometry selections.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]ANGCHECK.CPR;1

ANGCHECK:CPROC;
A=0.;
IF YCRD(ENDPOINT1(L)) ^= YCRD(ENDPOINT2(L)) THEN DO;
GETV ANGLE 'A' LINES (L) HORIZ CONSTR (P) REAL ACUTE;
END;
%END
```

B2.4.1 "CHARACTER"

Performs transformation of the floating point data type into the character string with the specified precision.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]CHARACTER.CPR;5

CHARACTER:CPROC;
DFL_1=DFL_2*10000;
FX_1=FLOOR(DFL_1);
CH32_1=CHARACTER(FX_1);
CH32_2=SUBSTR(CH32_1,11);
FX_2=FLOOR(DFL_1/10000);
CH32_3=CHARACTER(FX_2);
CH32_4=SUBSTR(CH32_3,11);
CH32_5='.';
CH32_6=CH32_4 !! CH32_5 !! CH32_2;
IF DFL_1=0 THEN CH32_6='0';
%END
```

B.2.4.2 "FWRITE"

Writes the variable into the external file.

```
SYS$SYSDEVICE:[ENG.NELIK.CPROCS]FWRITE.CPR;2

FWRITE:CPROC;
WRITE FILE(FILE1) FROM (CH32_6);
%END
```

B.3 Sample Output Files

FIGURE B.3.1

MACHINING FILE

```
$
$ TY 000001096.MCH
$ NCIMP
$ IMPMACH1
$ CAST # 3X10DACX1-00
$ 000001096
$ 62217773-00
$ METRIC
DVR2, 101.0000
DVR3,  93.1000
DVR4,   8.0000
DVR5,  70.9699
DVR6,  45.0000
DVR7,  91.9962
DVR8,  41.5220
DVR9, 138.0983
DVR10, 154.3002
DVR11, 161.9848
DVR14,  19.1422
DVR18,  24.0000
DVR188,  20.0000
DVR23,  28.0000
DVR24,   5.0000
DVR31,  45.0000
DVR32,   1.5000
DVR44,   5.0000
DVR45,  25.0000
DVR47,  76.1000
DVR55,  77.1000
DVR60,  16.0000
DVR61,0
DVR64,  45.0000
$ MANUAL INPUT AT CAD
DVR28,  10.0000
DVR29,   5.0000
DVR48,0
$ SUTTON: N
$ KEYWIDHT= .5
DVR63,   2.5000
DVR65,   6.2500
$
$
$
$
```

FIGURE B.3.2

CASTING FILE

```
$
$ TY 000000999.CAS
$ NCIMP
$ IMPCAST1
$
$ 000000999
$ 62216734-00
$ METRIC
DVR15,  23.9459
DVR16,  24.2218
DVR26, 394.1045
DVR50, 866.3500
DVR51,  27.3129
DVR53,  48.2178
DVR54,   7.7953
DVR57,  42.0000
$
```

APPENDIX C (AGL)

The programs listed here accomplish the same goals as the ones in the Appendix B. The programs are written, however, in the AGL Language.

It is advantageous to use these programs for the final production version, whereas the programs written in IAGL are preferred to be used in the development stage since their interactive nature allows more flexibility in debugging.

C.1 "APPMACH1" (Similar to IMPMACH1 of IAGL Routines)

```
SYS$SYSDEVICE:[ENG.NELIK.AGL]APPMACH1.AGL;31

APPMACH1:PROCEDURE;
%INCLUDE APPCXM;
%INCLUDE GMYENTRY;
%INCLUDE APPDBA;

/** EXTERNAL ENTRIES **/;
DCL APPCOIN  EXTERNAL ENTRY (POINT 3D,POINT 3D,POINT 3D,FIXED BIN(31) );
DCL APPDEL   EXTERNAL ENTRY (POINT 3D);
DCL APPPICKC EXTERNAL ENTRY (POINT 3D,CIRCLE 3D);
DCL APPPICKL EXTERNAL ENTRY (POINT 3D,LINE 3D);
DCL APPXENDC EXTERNAL ENTRY (CIRCLE 3D,POINT 3D,POINT 3D);
DCL APPXENDL EXTERNAL ENTRY (LINE 3D,POINT 3D,POINT 3D);
DCL APPXSAME EXTERNAL ENTRY (POINT 3D,POINT 3D);
DCL APPYENDC EXTERNAL ENTRY (CIRCLE 3D,POINT 3D,POINT 3D);
DCL APPYENDL EXTERNAL ENTRY (LINE 3D,POINT 3D,POINT 3D);
DCL APPYSAME EXTERNAL ENTRY (POINT 3D,POINT 3D);

DCL PA(30) POINT 3D;
DCL LA(23) LINE 3D;
DCL CA(6)  CIRCLE 3D;

DCL (I,ISW,EC) FIXED BIN(31);
DCL (DTYPEP,CLASSP,STRLENP,BYTESP) FIXED BIN(31);
DCL (DTYPEL,CLASSL,STRLENL,BYTESL) FIXED BIN(31);
DCL (DTYPEC,CLASSC,STRLENC,BYTESC) FIXED BIN(31);
DCL (XBASE,YBASE,R5) FLOAT BIN(53);
DCL (P,PS,PL,PS1,PL1,PS2,PL2) POINT 3D;
DCL (LHCL,LVCL,L1,L2) LINE 3D;
DCL C CIRCLE 3D;

/**INITILIZING CONSTANTS **/;
DTYPEP=-63; CLASSP=1; STRLENP=0; BYTESP=24;
DTYPEL=-61; CLASSL=1; STRLENL=0; BYTESL=48;
DTYPEC=-57; CLASSC=1; STRLENC=0; BYTESC=112;
```

```
SPEC DRAW PERL ON;
SPEC WORK MSGA 10; /**CHANGE FOR DEBUG **/;
SPEC WORK EXPAN ON; /**CHANGE FOR DEBUG **/;
SPEC WORK PICKL 8;

EC= GMY_DECLARE('P',DTYPEP,CLASSP,STRLENP);
EC= GMY_DECLARE('L',DTYPEL,CLASSL,STRLENL);
EC= GMY_DECLARE('C',DTYPEC,CLASSC,STRLENC);

MESS 'DIGITIZE HORIZONTAL C-LINE';
GETV LINE 'L' PICK NEAR ONE MAIN;
EC = GMY_GET_VALUE('L',LHCL);
CALL APPXENDL(LHCL,PS,PL);
CALL APPYSAME(PS,PL);
YBASE=YCRD(PL);

MESS 'DIGITIZE VERTICAL C-LINE';
GETV LINE 'L' PICK NEAR ONE MAIN;
EC = GMY_GET_VALUE('L',LVCL);
CALL APPYENDL(LVCL,PS,PL);
CALL APPXSAME(PS,PL);
XBASE=XCRD(PL);

MESS 'DIGITIZE STARTING LINE';
GETV LINE 'L' PICK NEAR ONE MAIN;
SPEC WORK PICKL 2; /** READY TO ROLL**/;
EC = GMY_GET_VALUE('L',LA(1));
CALL APPXENDL(LA(1),PS,PL);
CALL APPYSAME(PS,PL);

PA(1)=PS;
PA(2)=PL;
P=(PS+PL)/2;
EC = GMY_PUT_VALUE('P',DTYPEP,CLASSP,STRLENP,P);
CALL APPDEL(P);
CALL APPPICKC(PA(2),CA(1));
CALL APPXENDC(CA(1),PS,PL);
PA(3)=PL;

CALL APPDEL(PA(2));
CALL APPPICKL(PA(3),LA(2));
CALL APPYENDL(LA(2),PS,PL);
PA(4)=PL;

CALL APPDEL(PA(3));
CALL APPPICKL(PA(4),LA(3));
CALL APPXENDL(LA(3),PS,PL);
CALL APPYSAME(PS,PL);
PA(5)=PL;

CALL APPDEL(PA(4));
CALL APPPICKL(PA(5),LA(4));
CALL APPYENDL(LA(4),PS,PL);
PA(6)=PL;

CALL APPDEL(PA(5));
CALL APPPICKC(PA(6),CA(2));
CALL APPYENDC(CA(2),PS,PL);
PA(7)=PL;

CALL APPDEL(PA(6));
CALL APPPICKL(PA(7),LA(5));
CALL APPYENDL(LA(5),PS,PL);
PA(8)=PL;
```

```
CALL APPDEL(PA(7));
CALL APPPICKL(PA(8),LA(6));
CALL APPXENDL(LA(6),PS,PL);
CALL APPYSAME(PS,PL);
PA(9)=PL;

CALL APPDEL(PA(8));
CALL APPPICKL(PA(9),L1);
CALL APPYENDL(L1,PS1,PL1);
P=(PS1+PL1)/2;
CALL APPDEL(P);
CALL APPPICKL(PA(9),L2);
CALL APPYENDL(L2,PS2,PL2);
P=(PS2+PL2)/2;
CALL APPDEL(P);
IF YCRD(PS2) < YCRD(PS1) THEN
 DO;
  ADD LINE (L2);
  LA(7)=L1;
 END;
IF YCRD(PS2) > YCRD(PS1) THEN
 DO;
  ADD LINE (L1);
  LA(7)=L2;
 END;
CALL APPXENDL(LA(7),PS,PL);
CALL APPYSAME(PS,PL);
PA(10)=PL;

CALL APPPICKL(PA(10),L1);
CALL APPYENDL(L1,PS1,PL1);
P=(PS1+PL1)/2;
CALL APPDEL(P);
CALL APPPICKL(PA(10),L2);
CALL APPYENDL(L2,PS2,PL2);
P=(PS2+PL2)/2;
CALL APPDEL(P);
IF YCRD(PS2) < YCRD(PS1) THEN
 DO;
  ADD LINE (L2);
  LA(8)=L1;
 END;
IF YCRD(PS2) > YCRD(PS1) THEN
 DO;
  ADD LINE (L1);
  LA(8)=L2;
 END;
CALL APPXENDL(LA(8),PS,PL);
CALL APPYSAME(PS,PL);
PA(11)=PL;

CALL APPPICKL(PA(11),LA(9));
CALL APPYENDL(LA(9),PS,PL);
PA(12)=PS;

CALL APPDEL(PA(11));
CALL APPPICKC(PA(12),CA(3));
CALL APPYENDC(CA(3),PS,PL);
PA(13)=PS;

CALL APPDEL(PA(12));
CALL APPPICKL(PA(13),LA(10));
CALL APPYENDL(LA(10),PS,PL);
PA(14)=PS;

CALL APPDEL(PA(13));
```

```
CALL APPPICKL(PA(14),LA(11));
CALL APPXENDL(LA(11),PS,PL);
CALL APPYSAME(PS,PL);
PA(15)=PL;

CALL APPDEL(PA(14));
CALL APPPICKL(PA(15),LA(12));
CALL APPYENDL(LA(12),PS,PL);
PA(16)=PS;

CALL APPDEL(PA(15));
CALL APPPICKC(PA(16),CA(4));
CALL APPXENDC(CA(4),PS,PL);
PA(17)=PL;

CALL APPDEL(PA(16));
CALL APPPICKL(PA(17),LA(13));
CALL APPXENDL(LA(13),PS,PL);
CALL APPYSAME(PS,PL);
PA(18)=PL;

CALL APPDEL(PA(17));
CALL APPPICKL(PA(18),LA(14));
CALL APPXENDL(LA(14),PS,PL);
PA(19)=PL;

CALL APPDEL(PA(18));
CALL APPPICKL(PA(19),LA(15));
CALL APPYENDL(LA(15),PS,PL);
PA(20)=PS;

CALL APPDEL(PA(19));
 /** CHECKING PICKLIMITS **/;
I=0; ISW=0;
 DO WHILE (ISW ^= 1);
CALL APPPICKL(PA(20),L1);
CALL APPYENDL(L1,PS1,PL1);
CALL APPCOIN(PA(20),PS1,PL1,ISW);
I=I+1;
IF I>5 THEN MESS 'LINES END POINTS DO NOT COINCIDE';
IF I>5 THEN ABORT;
 END;
/***CONTINUES IF SUCCESFUL**/;
P=(PS1+PL1)/2;
CALL APPDEL(P);
 /** CHECKING PICKLIMITS **/;
I=0; ISW=0;
 DO WHILE (ISW ^= 1);
CALL APPPICKL(PA(20),L2);
CALL APPYENDL(L2,PS2,PL2);
CALL APPCOIN(PA(20),PS2,PL2,ISW);
I=I+1;
IF I>5 THEN MESS 'LINES END POINTS DO NOT COINCIDE';
IF I>5 THEN ABORT;
 END;
/**CONTINUES IF SUCCESSFUL**/;
P=(PS2+PL2)/2;
CALL APPDEL(P);
IF YCRD(PS2) < YCRD(PS1) THEN
 DO;
  ADD LINE (L2);
  LA(16)=L1;
 END;
IF YCRD(PS2) > YCRD(PS1) THEN
 DO;
  ADD LINE (L1);
```

```
  LA(16)=L2;
 END;
CALL APPXENDL(LA(16),PS,PL);
PA(21)=PS;

CALL APPPICKL(PA(21),L1);
CALL APPYENDL(L1,PS1,PL1);
P=(PS1+PL1)/2;
CALL APPDEL(P);
CALL APPPICKL(PA(21),L2);
CALL APPYENDL(L2,PS2,PL2);
P=(PS2+PL2)/2;
CALL APPDEL(P);

IF YCRD(PS2) < YCRD(PS1) THEN
 DO;
  ADD LINE (L2);
  LA(17)=L1;
 END;
IF YCRD(PS2) > YCRD(PS1) THEN
 DO;
  ADD LINE (L1);
  LA(17)=L2;
 END;
CALL APPXENDL(LA(17),PS,PL);
CALL APPYSAME(PS,PL);
PA(22)=PS;

/*** CHECKS PICKLIMITS ***/;
I=0; ISW=0;
 DO WHILE (ISW ^= 1);
CALL APPPICKL(PA(22),L1);
CALL APPYENDL(L1,PS1,PL1);
CALL APPCOIN(PA(22),PS1,PL1,ISW);
I=I+1;
IF I>5 THEN MESS 'LINES END POINTS DO NOT COINCIDE';
IF I>5 THEN ABORT;
 END;
/**CONTINUES IF SUCCESSFUL **/;
P=(PS1+PL1)/2;
CALL APPDEL(P);
/**CHECKING PICKLIMIT **/;
I=0; ISW=0;
DO WHILE (ISW ^= 1);
CALL APPPICKL(PA(22),L2);
CALL APPYENDL(L2,PS2,PL2);
CALL APPCOIN(PA(22),PS2,PL2,ISW);
I=I+1;
IF I>5 THEN MESS 'LINES END POINTS DO NOT COINCIDE';
IF I>5 THEN ABORT;
END;
/**CONTINUES IF SUCCESSFUL **/;
P=(PS2+PL2)/2;
CALL APPDEL(P);

IF YCRD(PS2) < YCRD(PS1) THEN
 DO;
  ADD LINE (L2);
  LA(18)=L1;
 END;
IF YCRD(PS2) > YCRD(PS1) THEN
 DO;
  ADD LINE (L1);
  LA(18)=L2;
 END;
CALL APPXENDL(LA(18),PS,PL);
```

```
PA(23)=PS;

CALL APPPICKL(PA(23),L1);
CALL APPYENDL(L1,PS1,PL1);
P=(PS1+PL1)/2;
CALL APPDEL(P);
CALL APPPICKL(PA(23),L2);
CALL APPYENDL(L2,PS2,PL2);
P=(PS2+PL2)/2;
CALL APPDEL(P);
IF YCRD(PS2) < YCRD(PS1) THEN
 DO;
  ADD LINE (L2);
  LA(19)=L1;
 END;
IF YCRD(PS2) > YCRD(PS1) THEN
 DO;
  ADD LINE (L1);
  LA(19)=L2;
 END;
CALL APPYENDL(LA(19),PS,PL);
PA(24)=PL;

CALL APPPICKL(PA(24),LA(20));
CALL APPXENDL(LA(20),PS,PL);
CALL APPYSAME(PS,PL);
PA(25)=PS;

CALL APPDEL(PA(24));
CALL APPPICKL(PA(25),L1);
CALL APPYENDL(L1,ps1,pl1);
P=(PS1+PL1)/2;
CALL APPDEL(P);
CALL APPPICKL(PA(25),L2);
CALL APPYENDL(L2,PS2,PL2);
P=(PS2+PL2)/2;
CALL APPDEL(P);
IF YCRD(PS2) < YCRD(PS1) THEN
 DO;
  ADD LINE (L2);
  LA(21)=L1;
 END;
IF YCRD(PS2) > YCRD(PS1) THEN
 DO;
  ADD LINE (L1);
  LA(21)=L2;
 END;
CALL APPYENDL(LA(21),PS,PL);
PA(26)=PL;

CALL APPPICKL(PA(1),LA(22));
CALL APPYENDL(LA(22),PS,PL);
PA(27)=PS;

CALL APPDEL(PA(1));
CALL APPPICKC(PA(27),CA(5));
CALL APPXENDC(CA(5),PS,PL);
PA(28)=PL;

CALL APPPICKL(PA(28),LA(23));
CALL APPXENDL(LA(23),PS,PL);
PA(29)=PL;

CALL APPPICKC(PA(29),CA(6));
CALL APPXENDC(CA(6),PS,PL);
```

```
/**ADDING DELETED GEOMETRY**/;
DO I=1 TO 22;
 ADD LINE (LA(I));
END;
DO I=1 TO 4;
 ADD CIRCLE (CA(I));
END;

/**GEOMETRY EXTRUCTIONS***/;
R5=RADIUS(CA(5));

/*****CALL APPMACH2(     )****/;

SPEC WORK PICKL 8;
RETURN;
END APPMACH1;

SYS$SYSDEVICE:[ENG.NELIK.AGL]APPMACH2.AGL;3
```

C.2 "IMPMACH2 - (Similar to IMPMACH2 of IAGL Routines)

```
IMPMACH2:PROCedure (pa,xbase,ybase,r5) ;
 dcl character external entry (float bin(53),char(50));

 dcl pa(30) point 3d;
DCL (DVR1,DVR2,DVR3,DVR4,DVR5,DVR6,DVR7,DVR8,
DVR9,DVR10,DVR11,DVR14,DVR18,DVR188,DVR23,
DVR24,DVR31,DVR32,DVR44,DVR45,DVR47,DVR55,
DVR60,DVR61,DVR64) FLOAT BIN(53);
DCL (df1_2,xbase,ybase,r5,DX,DY) FLOAT BIN(53);
DCL (ch32_6,R1,R2,R3,R4,R5A,R6) CHAR(50) VAR;
DCL (PARTNO) CHAR(50) VAR;
PARTNO='FDA0:[CADCAM]'!!STOREDB!!'.MCH';
R1='$ '!!RPART;
R2='$ '!!RMA_CA;
R3='$ CAST # '!!CASTDWGNO;
R4='$ '!!STOREDB;
R5A='$ '!!DRG_1;
R6='$ '!!INCH_MET;
/*****************************************/
```

```
/* DVR99 ALWAYS 1 RADIUS */
DVR1=(YCRD(P8)-YBASE)*2; /**ALWAYS MAX O.D. **/;
DVR2=(YCRD(P15)-YBASE)*2;
DVR3=(YCRD(P18)-YBASE)*2;
DVR4=XCRD(P19)-XCRD(P18);
DVR5=(YCRD(P21)-YBASE)*2;
DVR6=XCRD(P20)-XCRD(P25);
DVR7=XCRD(P20)-XCRD(P1);
DVR8=(YCRD(P26)-YBASE);
DVR9=(YCRD(P29)-YBASE)*2;
DVR10=(YCRD(P1)-YBASE)*2;
DVR11=(YCRD(P4)-YBASE)*2;
DVR14=XCRD(P10)-XCRD(P9);
/*DVR15 FROM CASTING */
/*DVR16 FROM CASTING */
DVR18=XCRD(P19)-XCRD(P15);
DVR188=XCRD(P4)-XCRD(P1);
/* DVR21 FROM CASTING */
DVR23=XCRD(P19)-XCRD(P13);
DVR24=R5;
/* DVR26 FROM CASTING */
/* DVR28 FROM RINGS */
/* DVR29 FROM RINGS */
/* DVR30 FROM ROUTING */
DVR31=XCRD(P19)-XCRD(P25);
DVR32=XCRD(P20)-XCRD(P21);
DVR44=XCRD(P24)-XCRD(P25);
DVR45=XCRD(P5)-XCRD(P1);
DVR47=(YCRD(P25)-YBASE)*2;
/* DVR48 FROM ROUTING  */
/* DVR49 DIFFERENCE IN DESIGN ! */
/* DVR50 FROM CASTING */
/* DVR51 FROM CASTING */
/* DVR53 FROM CASTING */
/* DVR54  FROM CASTING */
DVR55=(YCRD(P19)-YBASE)*2;
/* DVR57 FROM CASTING */
DVR60=XCRD(P30)-XCRD(P1);
DY=YCRD(P28)-YCRD(P29);
DX=XCRD(P29)-XCRD(P28);
DVR61=ATAND(DY/DX);
/* DVR63 FROM KEYWAY */
DY=YCRD(P18)-YCRD(P19);
DX=XCRD(P19)-XCRD(P18);
DVR64=ATAND(DY/DX);
/* DVR65 ENTERED .......*/
/* DVR72 ENTERED .......*/
```

```
/*********************************************/
VMS 'DELETE FDA0:[CADCAM]00000*.MCH;*' NONE;
CLOSE FILE(FILE1);
OPEN FILE(FILE1) TITLE(PARTNO) OUTPUT RECORD;
WRITE FILE(FILE1) FROM (R1);
WRITE FILE(FILE1) FROM (R2);
WRITE FILE(FILE1) FROM (R3);
WRITE FILE(FILE1) FROM (R4);
WRITE FILE(FILE1) FROM (R5A);
WRITE FILE(FILE1) FROM (R6);
DFL_2=DVR2; call character(dfl_2,ch32_6) ; CH32_6='DVR2,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR3; call character(dfl_2,ch32_6) ; CH32_6='DVR3,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR4; call character(dfl_2,ch32_6) ; CH32_6='DVR4,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR5; call character(dfl_2,ch32_6) ; CH32_6='DVR5,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR6; call character(dfl_2,ch32_6) ; CH32_6='DVR6,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR7; call character(dfl_2,ch32_6) ; CH32_6='DVR7,'!!CH32_6;
 write file(file1) from (ch32_6);
DFL_2=DVR8; call character(dfl_2,ch32_6) ; CH32_6='DVR8,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR9; call character(dfl_2,ch32_6) ; CH32_6='DVR9,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR10; call character(dfl_2,ch32_6) ; CH32_6='DVR10,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR11; call character(dfl_2,ch32_6) ; CH32_6='DVR11,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR14; call character(dfl_2,ch32_6) ; CH32_6='DVR14,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR18; call character(dfl_2,ch32_6) ; CH32_6='DVR18,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR188; call character(dfl_2,ch32_6) ; CH32_6='DVR188,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR23; call character(dfl_2,ch32_6) ; CH32_6='DVR23,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR24; call character(dfl_2,ch32_6) ; CH32_6='DVR24,'!!CH32_6;
 write file(file1) from (ch32_6);
DFL_2=DVR31; call character(dfl_2,ch32_6) ; CH32_6='DVR31,'!!CH32_6;
 write file(file1) from (ch32_6);
DFL_2=DVR32; call character(dfl_2,ch32_6) ; CH32_6='DVR32,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR44; call character(dfl_2,ch32_6) ; CH32_6='DVR44,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR45; call character(dfl_2,ch32_6) ; CH32_6='DVR45,'!!CH32_6;
 write file(file1) from (ch32_6);
DFL_2=DVR47; call character(dfl_2,ch32_6) ; CH32_6='DVR47,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR55; call character(dfl_2,ch32_6) ; CH32_6='DVR55,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR60; call character(dfl_2,ch32_6) ; CH32_6='DVR60,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR61; call character(dfl_2,ch32_6) ; CH32_6='DVR61,'!!CH32_6;
write file(file1) from (ch32_6);
DFL_2=DVR64; call character(dfl_2,ch32_6) ; CH32_6='DVR64,'!!CH32_6;
write file(file1) from (ch32_6);
/***** WRITING INFO FROM ROUTING ***/;
CH32_6='$ MANUAL INPUT AT CAD';
write file(file1) from (ch32_6); ;
/****call IMPROUT ;**/;
CLOSE FILE(FILE1);
return;
END appmach2;
```

C.3.1, C.3.2., APPXENDL, APPYENDL, APPXENDC, APPYENDC,

(Similar to XENDL, YENDL)

```
SYS$SYSDEVICE:[ENG.NELIK.AGL]APPXENDL.AGL;1

APPXENDL:PROCEDURE (L,PS,PL);
DCL (PS,PL,P) POINT 3D;
DCL L LINE 3D;
PS=ENDPOINT1(L);
PL=ENDPOINT2(L);
 IF XCRD(PL) < XCRD(PS) THEN DO;
  P=PS;
  PS=PL;
  PL=P;
 END;
RETURN;
END APPXENDL;
```

SYS$SYSDEVICE:[ENG.NELIK.AGL]APPYENDL.AGL;2

```
APPYENDL:PROCEDURE (L,PS,PL);
DCL (PS,PL,P) POINT 3D;
DCL L LINE 3D;
PS=ENDPOINT1(L);
PL=ENDPOINT2(L);
 IF YCRD(PL) < YCRD(PS) THEN DO;
  P=PS;
  PS=PL;
  PL=P;
 END;
RETURN;
END APPYENDL;
```

SYS$SYSDEVICE:[ENG.NELIK.AGL]APPXENDC.AGL;4

```
APPXENDC:PROCEDURE (C,PS,PL);
DCL (PS,PL,P) POINT 3D;
DCL C CIRCLE 3D;
PS=ENDPOINT1(C);
PL=ENDPOINT2(C);
 IF XCRD(PL) < XCRD(PS) THEN DO;
  P=PS;
  PS=PL;
  PL=P;
 END;
RETURN;
END APPXENDC;
```

```
SYS$SYSDEVICE:[ENG.NELIK.AGL]APPYENDC.AGL;3

APPYENDC:PROCEDURE (C,PS,PL);
DCL (PS,PL,P) POINT 3D;
DCL C CIRCLE 3D;
PS=ENDPOINT1(C);
PL=ENDPOINT2(C);
 IF YCRD(PL) < YCRD(PS) THEN DO;
  P=PS;
  PS=PL;
  PL=P;
 END;
RETURN;
END APPYENDC;
```

C.4.1, C.4.2., APPPICKL, APPPICKC (Similar to PICKL, PICKC of IAGL Routines

```
SYS$SYSDEVICE:[ENG.NELIK.AGL]APPPICKL.AGL;6

APPPICKL:PROCEDURE(P,L);
%INCLUDE GMYENTRY;
DCL (EC,DTYPEL,CLASSL,STRLENL) FIXED BIN(31);
DCL P POINT 3D;
DCL L LINE 3D;
GETV LINE 'L' PICK NEAR ONE MAIN (P);
EC = GMY_GET_VALUE('L',L);
RETURN;
END APPPICKL;
```

SYS$SYSDEVICE:[ENG.NELIK.AGL]APPPICKC.AGL;6

```
APPPICKC:PROCEDURE(P,C);
%INCLUDE GMYENTRY;
DCL EC FIXED BIN(31);
DCL P POINT 3D;
DCL C CIRCLE 3D;
GETV CIRCLE 'C' PICK NEAR ONE MAIN (P);
EC = GMY_GET_VALUE('C',C);
RETURN;
END APPPICKC;
```

C.5.1., C.5.6., APPXSAME, APPYSAME -(Similar to XSAME,
YSAME of IAGL Routines)

```
SYS$SYSDEVICE:[ENG.NELIK.AGL]APPXSAME.AGL;8

APPXSAME:PROCEDURE (PS,PL);
%INCLUDE GMYENTRY;
DCL (EC,DTYPEP,CLASSP,STRLENP) FIXED BIN(31);
DCL (PS,PL,PM,P) POINT 3D;
DTYPEP=-63; CLASSP=1; STRLENP=0;
PM=(PS+PL)/2;
 IF XCRD(PL) ^= XCRD(PS) THEN DO;
  MESS 'LINE IS NOT VERTICAL ! FIX IT AND RERUN.';
  EC =  GMY_PUT_VALUE ('P',DTYPEP,CLASSP,STRLENP,PM);
    LIST SELECT PICK NEAR ONE MAIN (P) ;
  SPEC WORK PICKL 8;
  ABORT;
 END;
RETURN ;
END APPXSAME;
```

SYS$SYSDEVICE:[ENG.NELIK.AGL]APPYSAME.AGL;11

```
APPYSAME:PROCEDURE (PS,PL);
%INCLUDE GMYENTRY;
DCL (EC,DTYPEP,CLASSP,STRLENP) FIXED BIN(31);
DCL (PS,PL,PM,P) POINT 3D;
DTYPEP=-63; CLASSP=1; STRLENP=0;
PM=(PS+PL)/2;
 IF YCRD(PL) ^= YCRD(PS) THEN DO;
  MESS 'LINE IS NOT HORIZONTAL ! FIX IT AND RERUN.';
  EC = GMY_PUT_VALUE ('P',DTYPEP,CLASSP,STRLENP,PM);
  LIST SELECT PICK NEAR ONE MAIN (P);
  SPEC WORK PICKL 8;
  ABORT;
 END;
RETURN ;
END APPYSAME;
```

C.6 APPDEL - (Similar to DELLI of IAGL Routines)

```
SYS$SYSDEVICE:[ENG.NELIK.AGL]APPDEL.AGL;3

APPDEL:PROCEDURE (P);
%INCLUDE GMYENTRY;
DCL (EC,DTYPEP,CLASSP,STRLENP) FIXED BIN(31);
DCL P POINT 3D;
 DTYPEP=-63; CLASSP=1; STRLENP=0;
EC = GMY_PUT_VALUE ('P',DTYPEP,CLASSP,STRLENP,P);
MODI DEL PICK NEAR ONE MAIN (P);
RETURN;
END APPDEL;
```

C.7. APPCOIN (Routine to determine to check if points have identical coordinates)

```
APPCOIN:PROCEDURE (PAT,PS,PL,ISW);
DCL (PAT,PS,PL) POINT 3D;
DCL ISW FIXED BIN(31);
DCL STRING CHARACTER(30) VAR;
DCL OUTFILE FILE;
PUT SKIP LIST ('IN APPCOIN...',PAT,PS,PL,ISW);
STRING = 'IN APPCOIN...';
ISW=0;
IF (XCRD(PAT)=XCRD(PS) ) & (YCRD(PAT)=YCRD(PS) )  THEN ISW=1;
IF (XCRD(PAT)=XCRD(PL) ) & (YCRD(PAT)=YCRD(PL) )  THEN ISW=1;
OPEN FILE(OUTFILE) RECORD OUTPUT TITLE('[ENG.NELIK.MISC]AGLFILE.DAT') ENV(APPEND);
WRITE FILE(OUTFILE) FROM (STRING);
WRITE FILE(OUTFILE) FROM (PAT);
WRITE FILE(OUTFILE) FROM (PS);
WRITE FILE(OUTFILE) FROM (PL);
WRITE FILE(OUTFILE) FROM (ISW);
CLOSE FILE(OUTFILE);
RETURN;
END APPCOIN;
```

12. VITA

The author was born in Izmail, U.S.S.R., on March 5, 1954. He was educated at Suvorov High School in Izmail, and graduated from Leningrad Polytechnic Institute in 1977 with the degree of Master of Science in Mechanical Engineering.

Following graduation, he worked for two years at Tashkent, U.S.S.R., as an engineer, designing pumps and turbines. In 1979, he arrived in the United States and accepted a position with the Ingersoll-Rand Company as a design engineer. He is actively involved in the total CAD/CAM efforts of Ingersoll-Rand, working to make the Factory of the Future a reality.

He is married to Wendy (Litrides). They have a son, Adam, and currently reside in Bethlehem, Pa.